Holger Bönicke

Flexible und plattformunabhängige Entwicklung mikrocontrollerbasierter mechatronischer Systeme für Nutzer ohne Vorwissen

Holger Bönicke

Flexible und plattformunabhängige Entwicklung mikrocontrollerbasierter mechatronischer Systeme für Nutzer ohne Vorwissen

Universitätsverlag Ilmenau
2013

Impressum

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Angaben sind im Internet über http://dnb.d-nb.de abrufbar.

Diese Arbeit hat der Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau als Dissertation vorgelegen.

Tag der Einreichung:	18. Februar 2013
1. Gutachter:	Univ.-Prof. Dr.-Ing. habil. Christoph Ament (Technische Universität Ilmenau)
2. Gutachter:	Univ.-Prof. Dipl.-Ing. Dr. med. (habil.) Hartmut Witte (Technische Universität Ilmenau)
3. Gutachter:	Dr.-Ing. Reinhard Pittschellis (Festo Didactic GmbH & Co. KG, Denkendorf)
Tag der Verteidigung:	2. Juli 2013

Technische Universität Ilmenau/Universitätsbibliothek
Universitätsverlag Ilmenau
Postfach 10 05 65
98684 Ilmenau
www.tu-ilmenau.de/universitaetsverlag

Herstellung und Auslieferung
Verlagshaus Monsenstein und Vannerdat OHG
Am Hawerkamp 31
48155 Münster
www.mv-verlag.de

ISBN 978-3-86360-069-3 (Druckausgabe)
URN urn:nbn:de:gbv:ilm1-2013000388

Titelfoto: photocase.com

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

ASSP	Anwendungsspezifische Standardprodukte
FPGA	Field Programmable Gate Arrays
IDE	Integrated Development Environment Deutsch: Integrierte Entwicklungsumgebung
LCD	Liquid Crystal Display Flüssigkristallanzeige
LED	Lichtemittierende Diode
MSR	Messen-Steuern-Regeln
µC	Mikrocontroller
µP	Mikroprozessor
PC	Personal Computer Deutsch: Einzelplatzrechner
PWM	Pulsweitenmodulation
UML	Unified Modeling Language Deutsch: Vereinheitlichte Modellierungssprache
SPI	Serial Peripheral Interface Deutsch: Serielle periphere Schnittstelle
SPS	Speicherprogrammierbare Steuerung

Kurzfassung

In dieser Arbeit werden Konzepte vorgestellt, mit deren Hilfe Entwicklungssysteme für mechatronische Systeme realisiert werden können. Die Ansätze gewährleisten auch für Nutzer ohne Vorwissen eine flexible und einfache Arbeit. In diesem Zusammenhang werden Ansätze vorgestellt, mit welchen die Funktionsweise der Sensoren, der Aktoren und des informationstechnischen Systems transparent dargestellt wird, so dass der Nutzer hierfür ein technisches Verständnis entwickeln kann.

Nach der Erläuterung der Grundlagen zur Arbeit und des Stands der Technik werden zunächst die Konzepte der mikrosystemtechnischen Entwicklungssysteme „EasyKit“ und „EasyKit macht Schule“ vorgestellt. Als informationstechnische Systeme werden hier Mikrocontroller eingesetzt, da diese bereits eine hohe funktionelle Integration besitzen und somit meist das Mittel der Wahl für unerfahrene Nutzer sind. Die elektronischen Schaltungen werden in Form von Hardwaremodulen bereitgestellt und mit modularen Softwarebausteinen graphisch programmiert. Dabei werden die Vorteile von Ablaufdiagrammen mit den Vorteilen von Datenflussplänen kombiniert, wodurch eine hohe Flexibilität erreicht wird. Nutzertests zeigten, dass auch Nutzer ohne technisches Vorwissen grundsätzlich mit dieser Art der Programmierung arbeiten können.

Dieses Konzept wurde anschließend erweitert, um die Programmierung weiter zu vereinfachen und zu flexibilisieren. Außerdem bestand das Ziel, auch die Hardwareentwicklung auf der Schaltungsebene für Nutzer ohne Vorwissen zu ermöglichen, um so Abstand von Ansätzen, bei welchen vorgefertigte Komponenten genutzt werden, nehmen zu können. Hierfür werden in der Arbeit Anforderungen formuliert und Ansätze für Lösungskonzepte vorgestellt. Der wichtigste Ansatz zur Flexibilisierung der Programmierung ist die Einführung einer neuen ergänzenden Programmierebene, auf welcher parallel mit graphischen und textuellen Methoden gearbeitet wird. Zur Unterstüt-

zung der Hardwareentwicklung auf der Schaltungsebene werden Ansätze vorgestellt, welche eine weitestgehend auf ihre Schnittstellenarten abstrahierte Modellierung von Sensoren und Aktoren innerhalb der Entwicklungsumgebung ermöglichen. So kann der Nutzer bei der Entwicklung der Anpassungsschaltungen zwischen den Sensoren, Aktoren und dem Mikrocontroller unterstützt werden. Außerdem können so Signale und Signalwandlungen an den Schnittstellen in der graphischen Oberfläche dargestellt und vom Nutzer nachvollzogen werden. Weitere Ansätze zur Vereinfachung der Entwicklung werden ebenso in der Arbeit vorgestellt. Die erfolgversprechendsten Ansätze wurden in eine Testumgebung implementiert und mit Mitgliedern der möglichen Nutzergruppen getestet.

Abstract

In this thesis new concepts for designing development systems for mechatronic systems are introduced. The concepts allow flexible and simple usage, even if the users don't have prior expert knowledge. For this purpose, approaches are presented, which allow a transparent illustration of the mode of operation of sensors, actuators and used platforms, allowing users to understand related technical topics.

In the first part of the thesis, basic knowledge and the state of the art are presented. After this, the concepts of the microsystems development systems "EasyKit" and "EasyKit macht Schule" are described. In these systems microcontrollers are used as platform, because they already contain a high functional integration. Because of this, novice users prefer to use them as platform of choice. The electronic circuits, including the microcontroller, are provided in shape modular hardware blocks. They are programmed graphically with modular software blocks. The approach of programming introduced, uses a combination of the advantages of sequential function charts and synchronous data flow charts which increases the flexibility. Tests showed that even users without prior technical knowledge were able to program the microcontroller with these languages.

The EasyKit concept was advanced, to offer increased flexibility and simplicity during programming. Besides, there was the goal to give users without expert knowledge the capability of developing electronics on the circuitry level, which is far more flexible than developing on the modular level. For this purpose, requirements are analyzed and new approaches are presented in the thesis. The most important approach, to make the software development more flexible, is the introduction of a new additional programming level, which supports graphical and textual programming methods at the same time. For assisting the user during the hardware development on the circuitry level, approaches are presented, which allow modeling most

sensors and actuators, by abstracting them to their types of interfaces. Through this, the user can be supported when developing driver circuits to be used between the sensors, actuators and the microcontroller. Besides, this approach allows a comprehensible visualization of the signal behavior and the signal transformation at the interface of the microcontroller. Further approaches to increase the usability during the development phase are also presented in the thesis.

The most promising approaches were implemented to a development environment and tested with members of the target audience.

Vorwort

Auch wenn eine Dissertation eine eigenständige Arbeit eines Autors ist, so sind gewöhnlich doch viele Personen aus dem Umfeld an der erfolgreichen Fertigstellung beteiligt. Die vorliegende Arbeit ist dabei natürlich keine Ausnahme. Daher möchte ich an dieser Stelle einige Worte über die wichtigsten beteiligten Personen loswerden.

Zuerst möchte ich natürlich meinem Doktorvater Christoph Ament danken. Er ermöglichte mir den Einstieg in die Thematik der modellbasierten Entwicklung, aus welcher der Grundgedanke für diese Arbeit entstand. Außerdem zeigten mir die vielen Gespräche und Diskussionen mit ihm immer wieder neue Sichtweisen auf, wodurch meine Arbeit sehr stark beeinflusst wurde.

Auch Mike Eichhorn möchte ich hiermit herzlich danken. Seit seiner Rückkehr an das Fachgebiet Systemanalyse verschaffte er mir durch seine Organisationsfähigkeiten ausreichend Freiraum, um meine Arbeit abschließen zu können. Auch war seine eigene Hingabe zur wissenschaftlichen Arbeit immer wieder ein Beispiel für mich, welches mich selbst enorm motivierte.

Aber auch die weiteren Kollegen des Fachgebiets Systemanalyse sowie des befreundeten Fraunhofer-Instituts für Optronik, Systemtechnik und Bildauswertung sollen hier erwähnt werden. Die vielen Fachgespräche ermöglichten die Lösung unterschiedlichster Probleme und halfen mir bei meiner fachlichen und persönlichen Weiterentwicklung. In diesem Rahmen möchte ich mich vor allem bei Peter Hilgers und Carsten Stiller bedanken, ohne deren Zusammenarbeit meine Zeit am Fachgebiet sicher kein so großer Erfolg für mich gewesen wäre. Auch die gemeinsamen Aktivitäten außerhalb der Universität waren für mich immer wieder aufbauend und motivierend.

Weiterhin geht mein Dank an Reinhard Pittschellis, welcher für mich nicht nur ein angenehmer Projektpartner im Projekt „EasyKit“ war. Die Diskussionen mit ihm halfen mir immer wieder Schwachstellen an meiner Arbeit und an den von mir entwickelten Konzepten aufzudecken und zu beseitigen.

Natürlich gilt auch meiner Familie ein großer Dank. Ohne die Unterstützung meiner Eltern wäre ich sicher nie an den Punkt gekommen, eine solche Arbeit beginnen zu können. Und auch während der Bearbeitung meiner Dissertation konnte ich mir immer sicher darüber sein, dass mir von ihnen der Rücken gestärkt wird, wenn dies nötig wäre. Auch meiner Frau Fernanda gilt an dieser Stelle mein Dank. Obwohl wir gleichzeitig an unseren Dissertationen arbeiteten und diese sogar fast gleichzeitig einreichten, fanden wir doch immer noch ausreichend Zeit, um mit dem Partner zu reden. Diese gegenseitige Unterstützung war für mich sehr wertvoll.

Meinen Freunden und Bekannten vom Volleyballteam der Sieben Zwerge sei hiermit ebenso gedankt. Die sportliche Aktivität, bei der ich regelmäßig meinen Kopf frei bekam und so neue Kraft für die Arbeit tankte, war für mich ein wichtiger Teil meiner Zeit in Ilmenau.

Schließlich möchte ich mich noch bei allen anderen Personen bedanken, welche ansonsten noch direkt oder indirekt an meiner Arbeit beteiligt waren. Hierzu gehören beispielsweise die Schüler, welche mich bei den Nutzertests unterstützten, oder die studentischen Hilfskräfte, welche mir bei der Realisierung einiger Testsysteme halfen. Die Namen aller dieser Personen hier aufzuführen, würde zu weit gehen. Jeder von ihnen, der diese Zeilen liest, wird jedoch wissen, dass er gemeint ist.

Vielen Dank!

1 Einleitung

In diesem Kapitel wird zunächst die Motivation zu dieser Arbeit beschrieben. Danach erfolgt die Formulierung der sich hieraus ergebenden Zielstellungen. Anschließend wird die Vorgehensweise vorgestellt, um diese Zielstellungen zu erreichen. Dabei wird auch der Aufbau dieser Arbeit beschrieben.

1.1 Motivation

Das Streben des Menschen, sich seine Arbeit zu erleichtern, ist eine Eigenschaft, welche sich bereits vor langer Zeit entwickelte. Hierfür wurden einfachste Werkzeuge entwickelt, welche im Laufe der Zeit durch immer komplexer werdende Maschinen ergänzt wurden. Zu Beginn des 20. Jahrhunderts übernahmen diese Systeme in erster Linie rein mechanische Aufgaben. Vor allem bei komplexeren Prozessen musste der Mensch dabei die Aufgabe der Regelung und Steuerung übernehmen. Mit dem Fortschreiten der Technik im Bereich der Elektronik und der Informationstechnik wurde es möglich, den Menschen in vielen Bereichen durch technische Regler zu ersetzen. Vor allem die Entwicklung flexibel parametrierbarer Regler wäre ohne die Weiterentwicklung der Informationstechnik gar nicht möglich gewesen. [Ise2008]

Durch diese Fortschritte wurde auch die Entwicklung mechatronischer Systeme vorangetrieben, was wiederum einen steigenden Bedarf an Entwicklern in diesem Bereich bewirkte. Eine wichtige Rolle hierbei spielen mechatronische Systeme, welche als informationstechnische Plattform einen Mikrocontroller verwenden. Die Wichtigkeit dieser Systeme zeigt sich darin, dass unser tägliches Leben von ihnen in Form verschiedenster Geräte bestimmt wird [Hea2002]. [Ise2008]

Es ist abzusehen, dass der Bedarf an Experten auf dem Gebiet der Mechatronik und der Informationstechnik schneller steigt als ihre Verfügbarkeit, was zu einem Fachkräftemangel führt [Kol2012]. Um diesen Bedarf decken zu können, wurden verschiedene Lösungen entwickelt.

Einige dieser Lösungen sehen vor, dass Entwickler bei ihrer Arbeit durch eine modellbasierte Entwicklung unterstützt werden, was die Bearbeitungszeit reduziert und auch komplexere Systeme beherrschbar macht [PDB2009]. Unter modellbasierter Entwicklung im Rahmen dieser Arbeit ist zu verstehen, dass der Anwender die gewünschte Funktionalität der zu entwickelnden Software mit Hilfe von Modellierungssprachen beschreibt, welche oft auch eine graphische Darstellung besitzen [BST2010]. Aus diesen Beschreibungen kann Quellcode generiert werden, so dass dem Anwender die eigentliche Programmierung mit textuellen Sprachen weitestgehend erspart bleibt. In Abschnitt 2.2.2 wird die Problematik der modellbasierten Entwicklung weiter diskutiert. Weitere Ansätze zur Deckung des steigenden Bedarfs an Entwicklern sehen vor, den Fachkräftenachwuchs durch die Steigerung der Technikbegeisterung von Schülern zu stimulieren. Dabei sollen die Schüler animiert werden, sich auch in ihrer Freizeit mit technischen Problemstellungen zu befassen. Auch hierfür werden modellbasierte Ansätze genutzt. Hierfür werden in Abschnitt 2.4 einige Beispiele vorgestellt.

Auch wenn in beiden Fällen modellbasierte Ansätze zum Einsatz kommen, weisen diese dennoch erhebliche Unterschiede auf, welche aus den jeweiligen Zielstellungen resultieren. Für die industrielle Anwendung muss eine hohe Flexibilität gewährleistet sein, welche nur durch die Entwicklung auf der Mikrocontrollerebene möglich ist. Allerdings wird hier davon ausgegangen, dass die Nutzer bereits über weitreichendes technisches Vorwissen verfügen. Für die Anwendung durch jüngere Nutzer stehen jedoch die Vereinfachung der Entwicklung und die Motivation im Vordergrund. Daher werden die Mikrocontroller in einen Nutzungskontext, wie zum Beispiel die Robotik, eingebettet. Das System wird mit Hilfe von vorgefertigten mechatronischen Funktionsmodulen aufgebaut und auf der Ebene dieser Module programmiert, so dass der Nutzer sich nicht um die hardwarenahe Elektronik- und Softwareentwicklung kümmern muss.

Die Arbeit mit solchen modulbasierten Systemen kann die Technikbegeisterung von jüngeren Nutzern wecken und so den Wunsch zur weiteren Nutzung im Hobby- und Freizeitbereich auslösen. Allerdings gelangen gerade diese Nutzer sehr schnell an die Grenzen solcher modulbasierter Systeme, da die Funktionalität des Gesamtsystems auf Kombinationen der einzelnen Module beschränkt ist. Im Hobby- und Freizeitbereich sollen jedoch vielfältige Aufgabenstellungen gelöst werden, wodurch hohe Anforderungen bezüglich der Flexibilität bestehen. Eine solche flexible Entwicklung ist nur auf der Ebene der Mikrocontroller möglich. Die Entwicklung auf dieser Ebene ist jedoch grundlegend verschieden zur Entwicklung mit modulbasierten Ansätzen, weshalb sich der Nutzer beim Umstieg von Grund auf mit dem Thema der Elektronik- und Softwareentwicklung beschäftigen muss. Dies ist im Hobby- und Freizeitbereich jedoch ein großes Problem, da die Nutzer sich hier meist ohne fachkundige Anleitung einarbeiten müssen. Dies stellt eine erheblich Einstiegsbarriere dar. Vor allem bei jüngeren Personen, welche ansonsten technikbegeistert sind, zeigt sich häufig eine Scheu, sich in eine so komplexe Thematik einzuarbeiten. Jedoch wäre gerade für diese jungen Leute das hierbei erworbene Wissen wichtig, da sie dieses später in fast allen technischen Bereichen einsetzen können. Vor allem würde auf diese Weise der Einstieg in die Nutzung von Entwicklungssystemen, welche für den Einsatz durch Experten vorgesehen sind, vereinfacht werden. An dieser Stelle setzt die vorliegende Arbeit an.

1.2 Zielstellung

Es werden Konzepte und Ansätze untersucht, welche den Einstieg in die Thematik der mikrocontrollerbasierten mechatronischen Systeme für technisch interessierte Nutzer ohne domänenspezifisches Expertenwissen unterstützen. Der Begriff domänenspezifisches Expertenwissen wird synonym mit dem Begriff Vorwissen genutzt und wird in Abschnitt 2.1.3 erklärt.

Die Ansätze sollen verschiedene Bedingungen erfüllen:

- Die Entwicklung der mechatronischen Systeme muss ohne Expertenwissen auf dem Gebiet der Elektronik- oder Softwareentwicklung möglich sein, um dem Nutzer die Angst vor der Komplexität der Thematik zu nehmen und um die Entwicklung möglichst einfach zu gestalten.
- Die Entwicklung muss auf der Ebene der Mikrocontroller und Schaltungen ermöglicht werden, so dass eine hohe Flexibilität in der Elektronik- und Softwareentwicklung gewährleistet ist. Nur so können die vielfältigen Aufgabenstellungen im Hobbybereich abgedeckt werden.
- Die Entwicklung muss weitestgehend plattformunabhängig erfolgen, um den Nutzern auch hier weitere Flexibilität zu ermöglichen.
- Das technische Verständnis über die Funktionsweise mikrocontrollerbasierter mechatronischer Systeme muss gefördert werden, um den Nutzer zum technischen Denken anregen zu können und ihm so die zukünftige Entwicklung mechatronischer Systeme auf einem professionellen Niveau zu erleichtern.

Ein auffälliger Zielkonflikt ist, dass die Entwicklung trotz der möglichst hohen Flexibilität weitestgehend ohne Expertenwissen erfolgen soll. Dies gilt sowohl für den Softwarebereich als auch für den Hardwarebereich. Das übliche Vorgehen zur Reduzierung des benötigten Expertenwissens ist die Modularisierung. Dabei werden Module bereitgestellt, in deren Entwicklung Expertenwissen bereits eingeflossen ist. Der Einsatz dieser Module ermöglicht zwar dann die Nutzung von Expertenwissen, ohne dieses selbst zu besitzen, jedoch schränkt er die realisierbare Funktionalität des Gesamtsystems ein.

Ziel der Arbeit ist es, für die Gestaltung von Systemen zur Entwicklung mikrocontrollerbasierter mechatronischer Systeme Anforderungen zu formulieren sowie Realisierungskonzepte zu entwickeln und diese analytisch und empirisch zu untersuchen. Dabei sollen die vorgenannten Bedingungen berücksichtigt werden, wobei vor allem die Zielerreichungskonflikte zwischen einfacher Nutzbarkeit und hoher Flexibilität im Zentrum stehen. In

diesem Rahmen müssen sowohl Aspekte der Hardware- als auch der Softwareentwicklung betrachtet werden.

Die aufgeführten Ziele und Rahmenbedingungen ergeben sich aus der zu untersuchenden Zielgruppe, welche aus Personen besteht, die sich in die Thematik der mikrocontrollerbasierten mechatronischen Systeme einarbeiten möchten. Im Normalfall können diese Personen nicht auf technisches Wissen auf den Gebieten der Elektronik und der Informationstechnik zurückgreifen. Lediglich Wissen, welches im Bereich der Schulausbildung vermittelt wird, kann vorausgesetzt werden. In den meisten Fällen wird es sich dabei um jüngere Menschen handeln, welche ein technisch orientiertes Hobby betreiben, in welchem mechatronische Systeme einsetzbar sind. Daher stehen vor allem diese jüngeren Personen im Zentrum der Untersuchungen dieser Arbeit. Das bedeutet nicht, dass ältere Menschen nicht berücksichtigt werden. Allerdings haben sich erwachsene technikbegeisterte Personen gewöhnlich bereits ein grundlegendes technisches Wissen angeeignet. Daher kommen sie zwar als Anwender in Frage, für die im Rahmen dieser Arbeit durchgeführten Untersuchungen sind sie jedoch ungeeignet.

1.3 Aufbau der Arbeit und Neuartigkeit

Der Aufbau der vorliegenden Arbeit ist in Abbildung 1 in einem Überblick dargestellt. Er entspricht dem chronologischen Vorgehen bei der Entwicklung und Untersuchung der neuen Konzepte.

Das folgende Kapitel beschäftigt sich mit den wichtigsten Grundlagen der Arbeit und dem Stand der Technik. Dabei wird zunächst erklärt, was unter dem Begriff der mikrocontrollerbasierten mechatronischen Systeme zu verstehen ist, wie diese entworfen werden und welche Besonderheiten bei der Entwicklung zu berücksichtigen sind. Dabei wird auch der Begriff des domänenspezifischen Expertenwissens definiert und grundlegende Informationen zur Mikrocontrollertechnik gegeben. Anschließend werden in Abschnitt 2.2 einige Grundlagen zum Thema der Programmierung von Mikrocontrollern vorgestellt. In diesem Rahmen werden auch die Begriffe der Plattformunabhängigkeit und der modellbasierten Softwareentwicklung diskutiert.

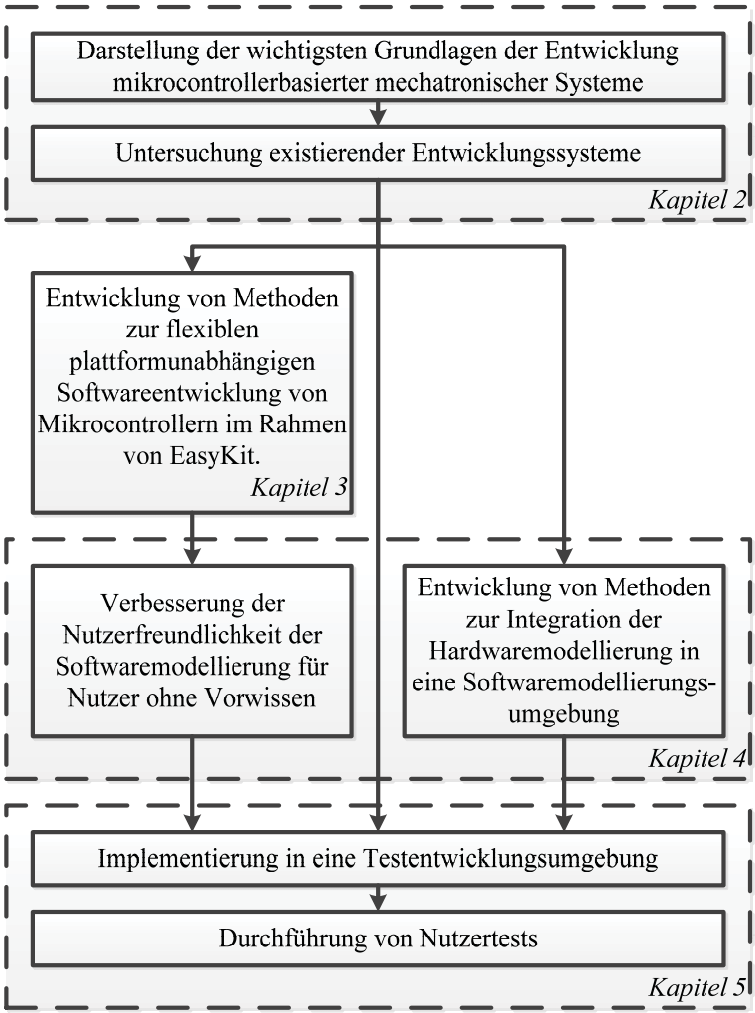

Abbildung 1: Aufbau dieser Arbeit

In Abschnitt 2.3 sind ausgewählte Grundlagen zum Thema der Mensch-Maschine-Schnittstelle dargestellt, wobei hier der Fokus auf der Analyse der Nutzerfreundlichkeit liegt. Im sich anschließenden Abschnitt 2.4 werden bereits existierende Systeme vorgestellt, welche mit Hilfe verschiedener Maßnahmen versuchen die Entwicklung mikrocontrollerbasierter mechatronischer oder verwandter Systeme für unterschiedliche Nutzergruppen zu erleichtern. Aus dieser Zusammenstellung wurden Defizite abgeleitet, welche in Abschnitt 2.5 diskutiert werden.

Im dritten Kapitel werden die ersten Lösungsansätze für die im zweiten Kapitel aufgeführten Defizite präsentiert. Dabei stand zunächst die Softwareentwicklung auf der Ebene der Mikrocontroller im Mittelpunkt, während für die Hardwareentwicklung weiterhin modulare Komponenten vorgesehen wurden. Als Ergebnis werden in den Abschnitten 3.1 und 3.2 die erste Version des Entwicklungssystems EasyKit und der für die Anwendung in der schulischen Ausbildung gedachte EasyKit Starter sowie die zugehörige Entwicklungsumgebung EasyLab vorgestellt. Mit dem EasyKit Starter wurden Nutzertests mit Experten, Lehrern und Schülern durchgeführt. Die Ergebnisse dieser Tests werden in Abschnitt 3.3 präsentiert. Des Weiteren erfolgt hier eine Diskussion der Defizite.

In Kapitel 4 werden Anforderungen formuliert, welche zur Beseitigung der Defizite herangezogen werden, um die Entwicklung des mechatronischen Systems verständlicher zu gestalten. Dabei wird nun auch bei der Hardwareentwicklung Abstand von modulbasierten Ansätzen genommen, um eine höhere Flexibilität zu erreichen. Für die Anforderungen werden verschiedene Realisierungskonzepte vorgeschlagen.

Im fünften Kapitel wird dargestellt, wie die auf Basis der formulierten Anforderungen entwickelten Realisierungskonzepte in eine Testentwicklungsumgebung implementiert wurden. In Abschnitt 0 werden die Durchführung und die Ergebnisse der mit der Testentwicklungsumgebung durchgeführten abschließenden Nutzertests diskutiert.

Das sechste und letzte Kapitel besteht aus der Zusammenfassung der Arbeit und einem Ausblick auf mögliche zukünftige Arbeiten.

Die Arbeit enthält folgende neuartige Ansätze:

- Die in dieser Arbeit vorgestellten Konzepte schließen die Lücke zwischen den einfach nutzbaren modulbasierten Ansätzen und der professionellen modellbasierten Entwicklung mechatronischer Systeme.
- Sie ermöglichen auch für Nutzer ohne Expertenwissen die Erfüllung der Forderungen einer plattformunabhängigen und flexiblen Entwicklung.
- Die im Rahmen der Vorstellung von EasyKit und des EasyKit Starters beschriebenen Konzepte wurden von Testnutzern unterschiedlicher Altersstufen als einfach nutzbar eingestuft und ermöglichen dennoch eine hohe Flexibilität in der Softwareentwicklung. Durch die Möglichkeit individuelle Schaltungen zu erstellen, ist das System erstmals sowohl durch Nutzer ohne Expertenwissen als auch durch Experten sinnvoll einsetzbar.
- In der Arbeit werden weitere neuartige Ansätze vorgestellt, welche die Entwicklung der Software soweit flexibilisieren und vereinfachen, dass ein Einsatz durch Nutzer im Hobbybereich ermöglicht wird.
- Weiterhin werden getestete Konzepte beschrieben, welche die Darstellung der elektronischen Hardware in der Entwicklungsumgebung ermöglichen.
- Es werden Ansätze aufgezeigt, welche eine schnelle Inbetriebnahme der informationstechnischen Plattform für Nutzer ohne Expertenwissen ermöglichen.
- Außerdem werden Konzepte vorgestellt, welche Nutzern ohne Expertenwissen die Entwicklung elektronischer Schaltungen auf Mikrocontrollerebene für die Anbindung von Sensoren und Aktoren ermöglichen.
- Es wird ein Programmierkonzept eingeführt, bei welchem der Nutzer auch ohne Expertenwissen einen hohen Freiheitsgrad erreicht. Dabei wird die Softwareentwicklung auf drei hierarchischen Ebenen durchgeführt.

- Dabei wird ein neuartiges hybrides Konzept aus graphischer und textueller Programmierung beschrieben, welches das Erlernen der grundlegenden Syntax einer textuellen Programmiersprache unterstützen kann.
- Weitere vorgestellte Ansätze der Arbeit machen dem Nutzer, unter anderem, bestimmte Funktion innerhalb des mechatronischen Systems, wie zum Beispiel der Signalwandlung am Mikrocontroller, verständlich. Das Verständnis dieser Funktionen kann den Einstieg in die professionelle Entwicklung mechatronischer Systeme erleichtern.

2 Grundlagen – Stand der Technik

In der Beschreibung der Zielstellung wurde ausgeführt, dass im Rahmen dieser Arbeit neue Ansätze für die Gestaltung von Systemen zur Entwicklung mikrocontrollerbasierter mechatronischer Systeme untersucht werden. Hierfür werden in erster Linie modellbasierte Ansätze berücksichtigt, welche genutzt werden, um die Plattformunabhängigkeit, Flexibilität und Einfachheit der Entwicklung zu gewährleisten.

In diesem Kapitel wird daher zunächst eine grundlegende Begriffsklärung zum Thema der mikrocontrollerbasierten mechatronischen Systeme vorgenommen. Anschließend wird die Programmierung von Mikrocontrollern erklärt, wobei auch Aspekte der Plattformunabhängigkeit, der modellbasierten Entwicklung und der graphischen Programmiersprachen diskutiert werden. Weiterhin werden Grundlagen zur Mensch-Maschine-Schnittstelle und Methoden zur Evaluierung der Nutzerfreundlichkeit dieser Schnittstellen vorgestellt. Im letzten Teil des Kapitels werden die Ergebnisse der Untersuchung bereits existierender Entwicklungssysteme für mikrocontrollerbasierte mechatronische Systeme vorgestellt.

2.1 Mikrocontrollerbasierte mechatronische Systeme

In diesem Abschnitt werden der Aufbau, die Funktionsweise und der Entwurf mikrocontrollerbasierter mechatronischer Systeme beschrieben. Hierfür wird zunächst der Begriff der mechatronischen Systeme erklärt. Anschließend wird der Einsatz von Mikrocontrollern in diesem Kontext diskutiert.

2.1.1 Mechatronische Systeme - Begriffsbestimmung

Die Mechatronik ist ein interdisziplinäres Fachgebiet, welches Elemente der Mechanik, der Elektronik und der Informatik in sich vereint. Dementsprechend sind mechatronische Systeme aus mechanischen, elektronischen und informationstechnischen Bestandteilen aufgebaut [Ise2008; Rod2012]. Die eigentliche, mit ihnen zu bearbeitende Aufgabenstellung ist jedoch meist im Bereich der Mechanik zu finden. Hierfür werden im mechatronischen System mechanische, chemische und optische Prozessgrößen zunächst mechanisch, chemisch oder optisch aufbereitet. Im nächsten Schritt erfolgt die Wandlung der physikalischen in elektrische Signale, so dass die Sensorelektronik diese anschließend in eine für das informationstechnische System nutzbare Repräsentation umwandeln kann. Im informationstechnischen System erfolgen Berechnungen, deren Ergebnisse wiederum zur Manipulation des Prozesses genutzt werden. Hierfür müssen die berechneten Daten zunächst wieder in elektrische Größen gewandelt werden, welche anschließend durch die Leistungselektronik an die benötigten Signalarten angepasst werden. Die so entstandenen elektrischen Signale werden wieder in nichtelektrische Größen umgewandelt und nach einer weiteren mechanischen, chemischen oder optischen Wandlung in den Prozess geführt. Grundsätzlich müssen nicht alle Schritte in genau dieser Form vorliegen. So entfällt beispielsweise die Signalaufbereitung durch nichtelektrische Messgrößenumformer, wenn die zu messende Prozessgröße eine elektrische ist. In Abbildung 2 ist der hier beschriebene Ablauf noch einmal im „Messen-Steuern-Regeln"-Kreislauf (MSR) graphisch dargestellt. [BST2010]

Der Nutzer kann das Verhalten des Systems auf zwei Arten beeinflussen. Zum einen kann er direkt in den Prozess eingreifen. Zum anderen kann er sich mit Hilfe des informationstechnischen Systems über den aktuellen Arbeitszustand des Prozesses und des mechatronischen Systems informieren und Einfluss nehmen, indem er die gewünschten Parameter in der Software entsprechend seiner Vorstellungen ändert. Dies ist natürlich nur möglich, wenn eine solche Einflussnahme am informationstechnischen System vorgesehen ist.

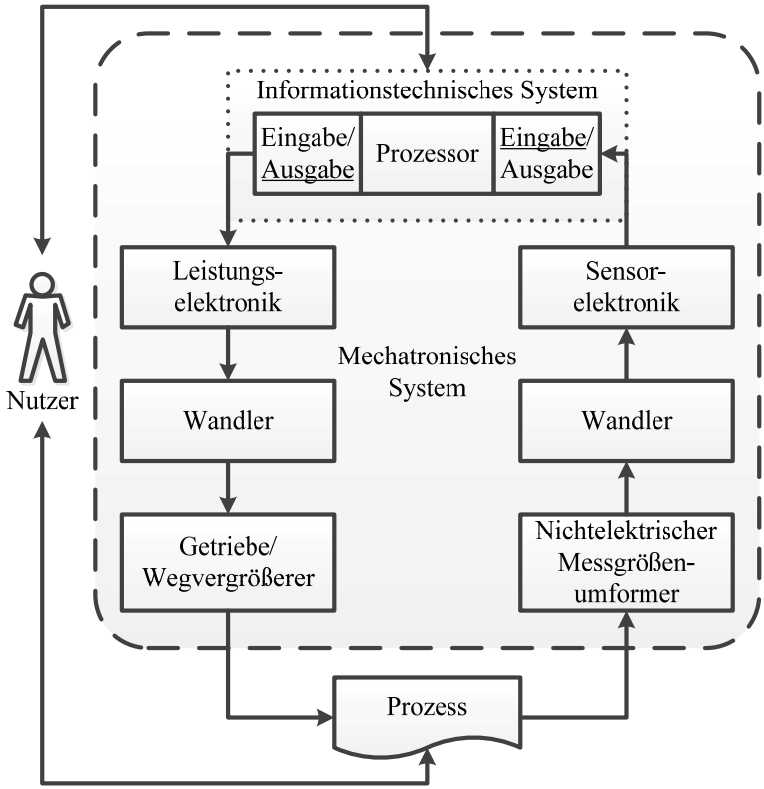

Abbildung 2: MSR-Kreislauf für ein mechatronisches System mit Einflussnahme durch einen Nutzer [BST2010; Jan2010; Val2010]

Entwurf mechatronischer Systeme

Da im Zentrum des Entwurfsprozesses die Funktionalität steht, existieren für die Realisierung der mechanischen, elektrischen und informationstechnischen Komponenten gewöhnlich viele verschiedene Möglichkeiten. Beim Entwurf eines mechatronischen Systems können diese nicht unabhängig voneinander und ebenso nicht unabhängig vom zugehörigen Prozess betrachtet werden. Somit entsteht ein komplexes Netzwerk aus Abhängigkei-

ten, was die Suche nach einer optimalen Lösung erschwert. Um Entwickler hierbei zu unterstützen, wurden modellbasierte Ansätze konzipiert. Dabei werden Verhaltensweisen und Wechselwirkungen der Prozesse und Teilsysteme beobachtet und modellhaft festgehalten. Auf Basis dieser Modelle wird anschließend das Gesamtsystem entwickelt. [Jan2010]

Häufig orientieren sich Entwickler bei der Durchführung einer solchen modellbasierten Entwicklung am V-Modell, welches in Abbildung 3 dargestellt ist. Auf der linken Seite des Modells erfolgt die Konkretisierung der einzelnen Komponenten, während auf der rechten Seite die Integration zum Gesamtsystem erfolgt. [Jan2010]

Modellbasierte Ansätze finden heute vor allem im Bereich der Softwareentwicklung Anwendung. In Abschnitt 2.2.2 wird hierauf noch einmal vertieft eingegangen.

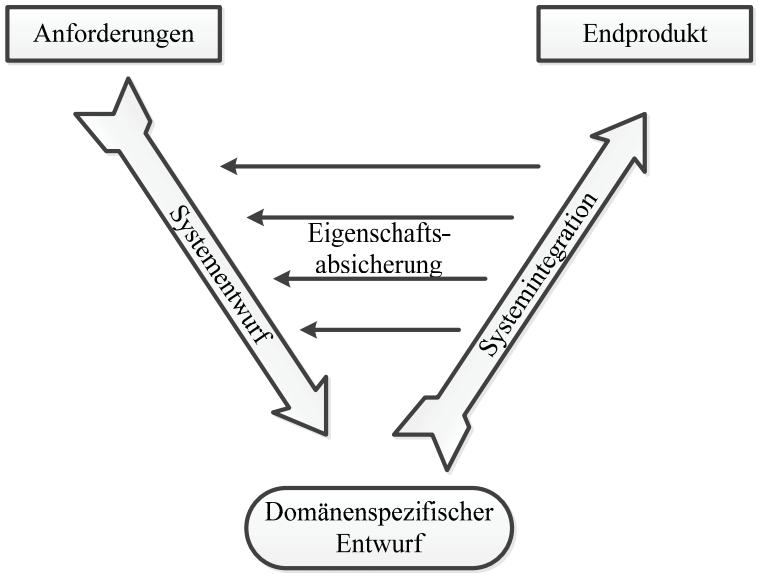

Abbildung 3: V-Modell zur modellbasierten Entwicklung [VDI2206-2004]

2.1.2 Mikrocontrollerbasierte mechatronische Systeme

Wird das informationstechnische System in einen spezifischen Nutzungskontext eingebunden, so spricht man von einem eingebetteten System [BST2010]. An solche Systeme werden oft spezifische Anforderungen gestellt. So müssen sie beispielsweise mobil, kostengünstig oder sehr spezialisiert sein. Es gibt sie in unterschiedlichen Konfigurationen, wobei zum Teil bereits Ein- und Ausgabeelektronik, Wandler und sogar mechanische Komponenten in das eingebettete System integriert sind [BST2010]. Die Hardware eingebetteter Systeme ist damit auf die Anforderungen für eine Integration in mechatronische Systeme ausgerichtet, was den Hauptunterschied zu herkömmlichen PC-Systemen ausmacht [BST2010; Ise2008].

Auch die Software des informationstechnischen Systems unterscheidet sich erheblich. Während herkömmliche PC-Systeme vom Nutzer durch die Installation neuer Programme in ihrer Funktionalität verändert werden können, ist das Umprogrammieren der Software in eingebetteten Systemen nur durch den Entwickler, nicht aber durch den Nutzer vorgesehen. Dieser soll lediglich die Möglichkeit zur Parametrierung der Software haben, was vollkommen ausreichend ist, da eingebettete Systeme, welche in größere, komplexere mechatronische Systeme integriert sind, nur bestimmte, vorher fest definierte Aufgaben übernehmen. [BST2010; Hea2002; LP2009; Pec2008; PSS2008]

Heutzutage werden vor allem vier verschiedene Typen von Entwicklungsplattformen genutzt, um eingebettete Systeme zu entwickeln [Dub2009]. In Tabelle 1 sind diese, gemeinsam mit ihren wichtigsten Eigenschaften, aufgeführt.

Im Rahmen dieser Arbeit werden Systeme betrachtet, welche als informationstechnisches System einen Mikrocontroller nutzen. Wie dies in Abbildung 4 dargestellt ist, besitzt ein Mikrocontroller bereits die wichtigste Peripherie, welche auch für den Betrieb eines von-Neumann-Rechners vorgesehen ist. Auf diese Weise wird die Integration in mechatronische Systeme erleichtert und die Entwicklungs- und Herstellungskosten können reduziert werden [Pec2008; Rod2012; Wüs2011].

Plattform	Eigenschaften
Mikroprozessor (µP)	Konfigurierbar über Software Für Rechenoperationen geeignet Bildet die Basiseinheit für die folgenden Plattformen
Mikrocontroller (µC)	Besitzt bereits integrierte Peripherie
Anwendungsspezifische Standardprodukte (ASSP)	Spezialisierte Peripherie Kann mit einem Hostprozessor kommunizieren
Field Programmable Gate Arrays (FPGA)	Möglichkeit die Vorteile von µP, µC und ASSP zu verbinden

Tabelle 1: Übersicht über die wichtigsten digitalen Entwicklungsplattformen [Dub2009]

Dennoch sind Mikrocontroller nicht so spezialisiert wie ASSPs oder FPGAs, so dass sie vergleichsweise flexibel einsetzbar sind und weniger spezialisiertes Fachwissen für die Nutzung benötigt wird [Cle1999; HB2009]. Durch die beiden Eigenschaften der flexiblen Einsetzbarkeit und der Kosteneffizienz erfüllen Mikrocontroller die wesentlichen Anforderungen, welche gewöhnlich an eingebettete Systeme gestellt werden [Mor2001].

Jedoch stehen den Vorteilen auch Nachteile entgegen. Vor allem ist hier zu nennen, dass die Verarbeitungsleistung von Mikrocontrollern vergleichsweise gering ist [Wüs2011].

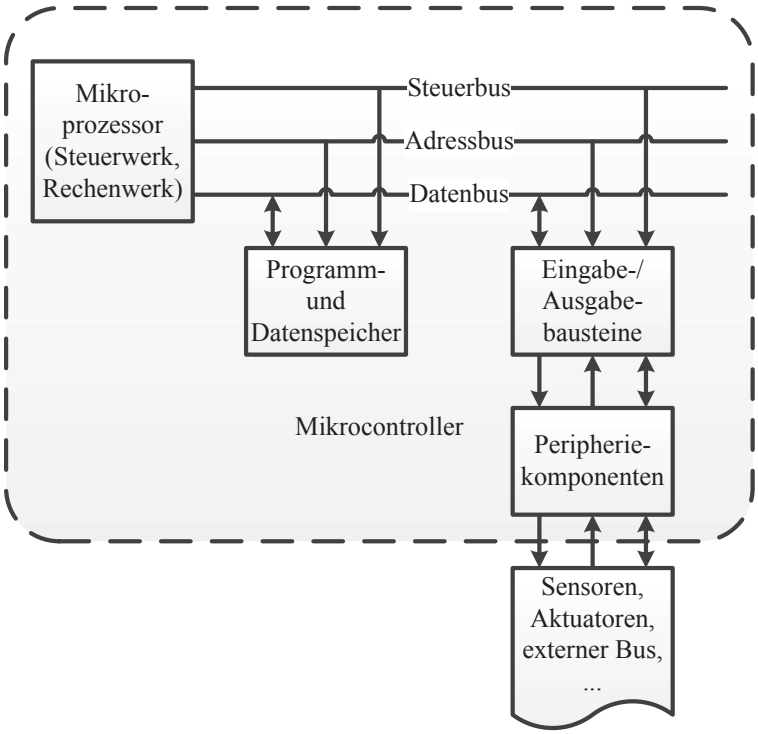

Abbildung 4: Darstellung eines Mikrocontrollers in Form eines von-Neumann-Rechners nach Wüst [Wüs2011]

Entwurf mikrocontrollerbasierter mechatronischer Systeme

Grundsätzlich kann für den Entwurfsprozess auf modellbasierte Ansätze, wie das in Abschnitt 2.1.1 beschriebene V-Modell, zurückgegriffen werden. Allerdings besitzen mikrocontrollerbasierte mechatronische Systeme folgende, zu berücksichtigende Besonderheiten:

- Mikrocontrollerbasierte mechatronische Systeme sollen meist stark spezialisiert, kostengünstig oder mobil sein [BST2010]. Die Auswahl der einsetzbaren mechanischen Komponenten reduziert sich somit.

- Zudem müssen die mechanischen Komponenten grundsätzlich mit den elektrischen Schnittstellen von Mikrocontrollern kompatibel sein.
- Die Entwicklung der Software muss an die Verarbeitungsleistung von Mikrocontrollern angepasst sein, da diese gegenüber anderen Plattformen eingeschränkt ist [Wüs2011].

Aufgrund dieser Besonderheiten werden mikrocontrollerbasierte mechatronische Systeme traditionell schrittweise entwickelt. In Abbildung 5 ist dies dargestellt [BPK2009].

Zunächst wird der mechanische Aufbau konstruiert, in welchen Sensoren und Aktoren integriert werden. Diese werden über Spannungsanpassungs- und Treiberschaltungen mit dem Mikrocontroller verbunden, so dass dieser die Sensordaten auslesen, verarbeiten und als Stellwerte an die Aktoren weiter geben kann.

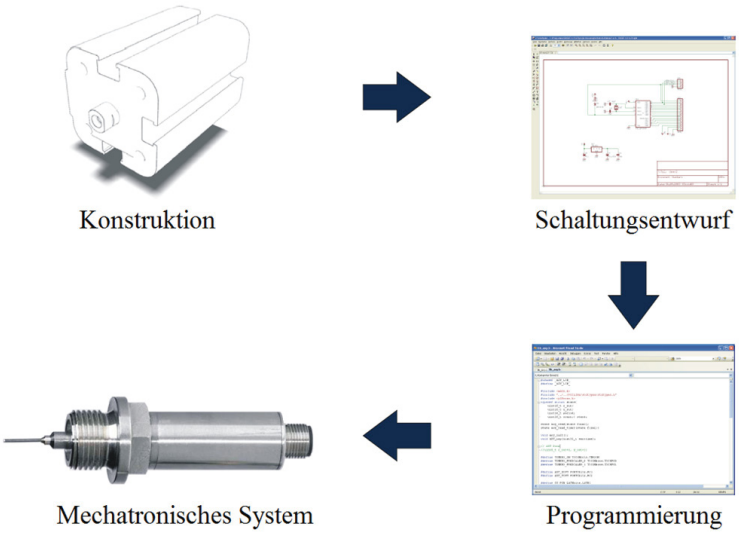

Abbildung 5: Sequentieller Ablauf der Entwicklung mechatronischer Systeme

Der Mikrocontroller muss im letzten Schritt programmiert werden. Der jeweilige Entwicklungsschritt baut auf den Ergebnissen des vorherigen Schrittes auf, so dass die Elektronik an die Mechanik und die Software an die Elektronik angepasst werden kann. Ein solcher sequentieller Arbeitsablauf wirkt sich daher positiv auf die Zuverlässigkeit des zu entwickelnden Systems aus [Ste2005].

Da die Entwicklung der Software der letzte Schritt im Systementwurf ist, muss hierbei die Kombination aus Prozess, nichtelektronischer Hardware, Elektronik und Software berücksichtigt werden, um so das entstehende Gesamtprodukt zu optimieren. Die Software muss beispielsweise an die Besonderheiten der genutzten Sensoren und Aktoren angepasst werden und gleichzeitig Randbedingungen, wie zum Beispiel Echtzeitfähigkeit, geringer Ressourcenbedarf, geringer Energieverbrauch oder eine hohe Zuverlässigkeit berücksichtigen. [BST2010]

2.1.3 Domänenspezifisches Expertenwissen im Bereich der mikrocontrollerbasierten mechatronischen Systeme

Wie in Abschnitt 2.1.2 beschrieben, wird bei der Entwicklung mikrocontrollerbasierter mechatronischer Systeme domänenspezifisches Expertenwissen aus unterschiedlichen Fachgebieten benötigt. Aus diesem Grund wird im Folgenden eine allgemeine Erklärung des Begriffs des Expertenwissens vorgenommen und anschließend auf das Problem der mikrocontrollerbasierten mechatronischen Systeme erweitert.

Domänenspezifisches Wissen im Allgemeinen

Als domänenspezifisches Expertenwissen oder Vorwissen werden Kenntnisse und Fertigkeiten von Personen auf bestimmten Gegenstandsgebieten bezeichnet. [Ren1996]

Domänenspezifisches Expertenwissen auf Grundlagengebieten erleichtert oder ermöglicht oft das Erlernen neuen Wissens aus anderen Domänen. [Ren1996]

Allerdings kann Expertenwissen auch zu Missverständnissen führen, wenn beispielsweise ein Begriff in zwei unterschiedlichen Domänen verschiedene Bedeutungen besitzt [NB1999]. Ein anschauliches Beispiel hierfür wird in

Abschnitt 2.2.2 unter dem Punkt „Abgrenzung zur modellbasierten Entwicklung in der Regelungstechnik“ vorgestellt. Hier wird kurz beschrieben, welche unterschiedlichen Bedeutungen der Begriff der modellbasierten Entwicklung im Bereich der Informatik und im Bereich der Regelungs- und Automatisierungstechnik besitzen.

Domänenspezifisches Wissen im Bereich der Mechatronik

Vor allem auf kombinierten Wissensgebieten, wie der Mechatronik, ist das Vorhandensein von Domänenwissen in den relevanten Teilbereichen wichtig, um neues Wissen möglichst einfach erlernen zu können. Als relevante Teilgebiete sind dabei unter anderem folgende Bereiche zu nennen [BH2011]:

- Mechanik
  - Mechanische, chemische und optische Prozesse (je nach Einsatzgebiet)
  - Sensorik
  - Aktorik
- Elektronik, Elektrotechnik
  - Schaltungsentwicklung
  - Signalwandlung
- Informationstechnik
  - Informationsdarstellung (Signale und Datentypen)
  - Softwareentwicklung

Je nach Einsatzbereich ist Vorwissen aus weiteren Fachgebieten, wie zum Beispiel der Kommunikations-, Automatisierungs- oder Kraftfahrzeugtechnik notwendig. [BH2011]

Arbeiten ohne domänenspezifisches Expertenwissen

Durch die im Rahmen dieser Arbeit entwickelten Konzepte sollen Nutzer mikrocontrollerbasierte mechatronische Systeme entwickeln können, ohne dabei Vorwissen auf den oben genannten Teilgebieten besitzen zu müssen. Um dies zu ermöglichen stehen grundsätzlich zwei Methoden zur Auswahl:

- Abstraktion durch Modulbildung:
  Das für die Erfüllung der Aufgabenstellung benötigte Expertenwissen ist bereits in modularen Systembausteinen integriert. Der Nutzer arbeitet mit den Modulen auf einer höheren abstrakten Ebene und greift somit indirekt auf das Expertenwissen der unteren Ebenen zurück, ohne selbst Experte auf dem Gebiet sein zu müssen.
- Hilfestellung durch zusätzliche Informationen:
  Das relevante Expertenwissen wird passend für die Nutzergruppe aufbereitet. Der Nutzer wird mit genau den Informationen versorgt, welche er für die Erfüllung seiner Aufgabenstellung benötigt.

Natürlich sind auch Kombinationen der beiden Methoden vorstellbar, so dass ihre jeweiligen Vorteile kombiniert werden können.

Bei beiden Herangehensweisen besteht allerdings das Problem, dass zunächst das benötigte Expertenwissen ausreichend genau definiert sein muss. Hierfür müssen die bestehenden Aufgabenstellungen und ihre zugehörigen Lösungen untersucht und auf das Vorwissen der späteren Nutzergruppe angepasst werden. In Abschnitt 2.3.5 werden hierfür anwendbare Untersuchungsmethoden beschrieben.

2.2 Programmieren von Mikrocontrollern

Um die eigentliche Programmierung der Software eines mikrocontrollerbasierten mechatronischen Systems nachvollziehen zu können, bietet sich eine Betrachtung der Programmiersprache C an. C wird als „High-Level Language“ bezeichnet. Auf der einen Seite gibt es zwar höhere Sprachen, wie C++, Java oder von modellbasierten Ansätzen abgeleitete graphische Pro-

grammiersprachen, welche immer mehr an Bedeutung gewinnen und daher später hier noch vorgestellt werden. Allerdings ist C bis heute noch die meist genutzte Programmiersprache für Mikrocontroller [BST2010]. Hinzu kommt, dass bei der Entwicklung mit höheren und graphischen Programmiersprachen meist aus dem Programm zunächst C-Code erzeugt wird, welcher danach wie ein ganz normales C-Programm weiterverarbeitet wird [BGB2008; Mat2012a].

Übersetzen und Herunterladen von C-Code auf einen Mikrocontroller

Ein Mikrocontroller kann nicht mit dem reinen C-Code arbeiten. Daher muss dieser mit Hilfe einer Werkzeugkette (oder engl. „Toolchain“) übersetzt werden. In Abbildung 6 ist der im Folgenden beschriebene Ablauf graphisch dargestellt.

Zunächst wird der C-Code durch einen Vorprozessor für den Compiler vorverarbeitet. Der Compiler erzeugt aus diesem vorverarbeiteten Code einen plattformspezifischen Code in einer Assemblersprache. Dieser Code besteht aus nacheinander ablaufenden Schritten, mit welchen direkt auf die Register des Mikroprozessors zugegriffen werden kann. Zum Teil werden einfache Programme für Mikroprozessoren direkt in dieser Sprache programmiert, da der Code so besser auf das System optimiert werden kann. Allerdings ist ein so erstellter Quellcode vergleichsweise unübersichtlich, weshalb diese Sprache heute nur noch selten genutzt wird. Die einzelnen Schritte werden im Folgenden durch den Assembler in eine bitcodierte Maschinensprache übersetzt, welche vom Mikroprozessor gelesen werden kann. Auch wenn der Code nun eigentlich lesbar für Mikroprozessoren wäre, kann das Programm so noch nicht ausgeführt werden, da den Variablen und Datenstrukturen bis zu diesem Punkt noch keine Speicheradressen zugewiesen wurden. Dies geschieht durch den Linker und den Loader. Der Linker kann zudem zusätzliche Bibliotheken einbinden. [Pec2008]

Diese Bibliotheken werden in vorkompilierter Form von Herstellern zur Verfügung gestellt, was in erster Linie dem Schutz geistigen Eigentums dient. Außerdem wird auf diese Weise die Übersichtlichkeit und Handhabbarkeit des Quellcodes verbessert. Im letzten Schritt muss das Programm in den Speicher der Plattform geladen werden. [Pec2008]

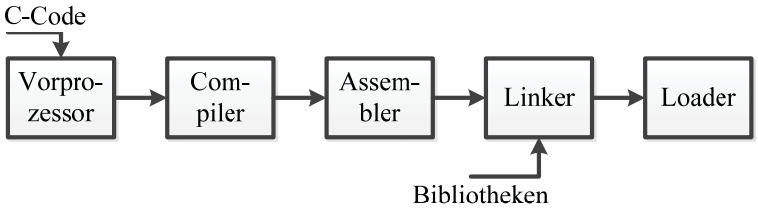

Abbildung 6: Übersetzung von C-Code zu Maschinencode [Pec2008]

2.2.1 Plattformunabhängigkeit mikrocontrollerbasierter mechatronischer Software

Eine wichtige Besonderheit von Software für Mikrocontroller ist, dass diese speziell für eine bestimmte Plattform entwickelt wird. Aus diesem Grund kann sie auch in die Plattformsoftware und die Applikationssoftware unterteilt werden. Die Plattformsoftware ermöglicht die Nutzung der von der Hardware zur Verfügung gestellten Funktionalität. Sie beinhaltet beispielsweise Treiber zur Kommunikation oder zur Interruptverarbeitung. Man kann also sagen, dass die Plattformsoftware festlegt, über welche hardwarenahen Funktionen die Arbeitsaufgaben des eingebetteten Systems realisiert werden sollen. In der Applikationssoftware hingegen wird festgelegt, welche Arbeitsaufgaben vom Mikrocontroller ausgeführt werden. Diese Arbeitsaufgaben, wie zum Beispiel das Abarbeiten von Regelalgorithmen, werden oft weitestgehend hardwareunabhängig implementiert. [BST2010]

Plattformunabhängigkeit - Begriffsbestimmung

Unter Plattformunabhängigkeit versteht man gewöhnlich, dass eine Software auf unterschiedlichen Plattformen ausgeführt werden kann. Als Plattformen können dabei die eigentliche Computerhardware, Betriebssysteme oder auch Dienstprogramme angesehen werden. Für ein herkömmliches Computerprogramm bedeutet dies beispielsweise, dass es, um plattformunabhängig zu sein, auf unterschiedlicher Hardware und unterschiedlichen Betriebssystemen lauf- und funktionsfähig sein muss.

Es existieren unterschiedliche Ausprägungen der Plattformunabhängigkeit. Die wichtigsten sind:

- Quellcodeunabhängigkeit (wichtig für Entwickler)
  - Plattformunabhängiger Quellcode kann für unterschiedliche Betriebssysteme übersetzt und auf diesen ausgeführt werden.
- Unabhängigkeit durch Nutzung eines Dienstprogramms
  - Das Programm wird in einem Dienstprogramm ausgeführt. Beispielsweise können Webanwendungen in einem Browser ausgeführt werden, unabhängig davon, auf was für einer Plattform dieser installiert ist.
- Zwischencode
  - Das Programm liegt in einer teilweise übersetzten Codeform vor, welche von einer Laufzeitumgebung interpretiert werden kann. Ein Beispiel hierfür ist Java, dessen Bytecode in der Java Laufzeitumgebung [Ora2012] ausgeführt wird.

Plattformunabhängige Softwareentwicklung für Mikrocontroller

Im Bereich der herkömmlichen Computersoftware existieren bereits viele Werkzeuge, mit welchen die oben genannten Plattformunabhängigkeiten ermöglicht werden können. Bei der Programmierung von Mikrocontrollern gibt es hingegen erst einige wenige Ansätze. In höheren textuellen Sprachen, wie C, C++ oder Embedded Java, ist dies bedingt möglich. Dabei muss der Entwickler selbst Maßnahmen treffen, dass im Quellcode keine plattformspezifischen Anweisungen vorkommen.

Spätestens wenn jedoch die Hardwareperipherie des Mikrocontrollers genutzt werden soll, ist es nahezu unmöglich, vollständig plattformunabhängig zu arbeiten, da unterschiedliche Mikrocontroller im Allgemeinen auch unterschiedliche Peripherie besitzen. Die Nutzung der Hardwareperipherie wird beispielsweise notwendig, sobald Sensordaten erfasst oder angeschlossene Aktoren manipuliert werden sollen. Da dies eine der Hauptaufgaben von Mikrocontrollern ist, kann man davon ausgehen, dass textuell erzeugter Quellcode für Mikrocontroller nicht ohne weitere Anpassungen oder zusätz-

liche Bibliotheken auf beliebige andere Plattformen übertragen werden kann.

Als Beispiel dient das Auslesen eines Analog-Digital-Wandlers eines Mikrocontrollers. Der Entwickler erstellt hierfür eine Bibliothek, in welcher er das Auslesen des Analog-Digital-Wandlers implementiert. Das Programm des Entwicklers kann nun mit Hilfe eines abstrakten Befehls auf die Bibliothek zugreifen. Möchte der Entwickler sein Programm auf einen anderen Mikrocontroller portieren, so muss auch hier eine entsprechende Bibliothek mit einer gleichlautenden Anweisung existieren, welche auf den Analog-Digital-Wandler des neuen Mikrocontrollers zugreift. Natürlich muss der Mikrocontroller hierfür über einen solchen Wandler verfügen. Ähnliches wie für die textuellen Sprachen gilt natürlich auch für die graphischen.

Daher wird im Rahmen dieser Arbeit unter dem Begriff der Plattformunabhängigkeit von Software nicht die uneingeschränkte Portierbarkeit des erzeugten Quellcodes auf andere Mikrocontroller verstanden. Vielmehr soll der Nutzer, trotz der unterschiedlichen Eigenschaften der Plattformen, fähig sein, diese mit einheitlichen Methoden zu programmieren. Beim Wechsel der Plattform kann damit der Großteil der erstellten Funktionen der Applikationssoftware weiterverwendet werden, so dass lediglich Anpassungen der Plattformsoftware notwendig sind.

2.2.2 Modellbasierte Softwareentwicklung

An Software für mikrocontrollerbasierte mechatronische Systeme werden hohe Anforderungen gestellt, wie zum Beispiel Echtzeitfähigkeit, geringer Ressourcen- und Energiebedarf oder eine hohe Zuverlässigkeit. Die Einhaltung der genannten Anforderungen wird heutzutage nicht nur durch die Beschäftigung ausgebildeter Softwareentwickler sichergestellt, sondern auch durch die Verbesserung des Softwareentwicklungsprozesses selbst. Bestand der Applikationsentwurf anfangs lediglich aus einer kurzen Anforderungsanalyse, deren Ergebnisse dann direkt in Assembler oder C umgesetzt wurden, werden heute oft Konzepte für eine modellbasierte Softwareentwicklung eingesetzt. Diese sehen vor, dass der Entwickler mit Hilfe von Modellierungssprachen nur noch die gewünschte Funktionalität der zu entwickelnden Software beschreibt. Die so erstellten Modelle können weitergenutzt

werden, um beispielsweise den eigentlichen Quellcode zu generieren. [BST2010]

Modellbasierte Ansätze wurden zunächst nur für die Hardwareentwicklung eingesetzt. Dabei wird die Funktionalität der Hardware auf der Ebene von Transistoren, Gates, Registern oder Systemen beschrieben [GVN1994]. Auch bei der modellbasierten Softwareentwicklung haben sich mittlerweile verschiedene Modellierungstechniken mit unterschiedlichen Darstellungsformen etabliert. Durch die Nutzung visueller Softwarebeschreibungssprachen kann die Funktionalität der zu entwickelnden Software auf unterschiedlichen Ebenen dargestellt werden. Einige dieser Sprachen wurden in graphische Entwicklungsumgebungen implementiert, um so Modelle einfacher entwickeln zu können. Meist kann der eigentliche Quellcode von der Entwicklungsumgebung direkt aus diesen Modellen generiert werden. [BST2010]

Der Vorteil eines solchen Vorgehens ist, dass die Komplexität der eigentlichen Programmierung verringert werden kann, was vor allem bei komplexeren Systemen ein deutlicher Vorteil ist. Vor allem Funktionen wie Regler und Filter, welche in textuellen Programmiersprachen oft unübersichtlich viele Quellcodezeilen ergeben, können in wenigen Schritten implementiert werden. Durch diese Verringerung der Komplexität erhöht sich die Zuverlässigkeit des Systems. Zudem kann häufig Entwicklungszeit gespart werden. [BST2010]

Allerdings stehen diesen Vorteilen auch Nachteile entgegen. Da die in der modellbasierten Entwicklung genutzten Funktionen für ein breites Feld an Anwendungsfällen konzipiert sein müssen, entsteht durch ihre Implementierung ein nicht auf die Aufgabenstellung optimierter Quellcode. Auch wenn die Entwicklung der Codegeneratoren voranschreitet, so ist es oft notwendig, den Quellcode manuell für die Anforderungen zu optimieren und an die Hardware anzupassen. Des Weiteren müssen im System viele verschiedene Funktionen zur Verfügung stehen, um ein breites Aufgabenspektrum abzudecken. Hier kann es zum einen dazu kommen, dass eine gewünschte Funktionalität gar nicht in dieser Form umsetzbar ist oder dass das Vorhandensein zu vieler unterschiedlicher Funktionen die Komplexität der Entwicklung wiederum erhöht. [BST2010]

Die modellbasierte Softwareentwicklung setzt sich aufgrund der bereits erwähnten Weiterentwicklung der Codegeneratoren immer weiter durch. Vor allem in Bereichen wie der Automobilindustrie, in welchen trotz hoher Komplexität die Qualität der Software gewährleistet sein muss, wird bereits intensiv mit modellbasierten Techniken gearbeitet. [BST2010]

Abgrenzung zur modellbasierten Entwicklung in der Regelungstechnik

Mikrocontroller werden in mechatronischen Systemen oft eingesetzt, um regelungstechnische Aufgaben auszuführen. Da Mikrocontroller vergleichsweise einfach zu nutzen sind, werden solche Systeme häufiger von Maschinenbauern und Regelungstechnikern programmiert als von ausgebildeten Informatikern. Allerdings ist gerade in der Regelungstechnik der Begriff der modellbasierten Entwicklung mit einer anderen Bedeutung verknüpft, weshalb hier kurz auf diesen Punkt eingegangen wird.

In der Regelungstechnik werden Modelle genutzt, um reale Prozesse vereinfacht nachzubilden. Auf Basis dieser Modelle werden Regler entwickelt, welche ein gewünschtes Verhalten des Gesamtsystems bewirken.

Beim modellbasierten Softwareentwurf wird nicht der eigentliche Prozess modelliert, sondern lediglich die Funktionalität der Software. Diese wird somit für den Entwickler vereinfacht und übersichtlich dargestellt.

2.2.3 Graphische Softwaremodellierung und Programmierung mikrocontrollerbasierter mechatronischer Systeme

Als graphische oder visuelle Programmierung wird die Softwareerstellung mit Hilfe graphischer Softwarebeschreibungssprachen und graphischer Programmiersprachen bezeichnet [Sch1998].

In graphischen Softwarebeschreibungssprachen werden verschiedene Aspekte der zu entwickelnden Software mit Hilfe graphischer Sprachen beschrieben. Graphische Programmiersprachen sind graphische Sprachen, mit welchen die Eigenschaften der zu entwickelnden Software vollständig beschrieben werden. Grundsätzlich kann aus beiden Beschreibungsformen Quellcode generiert werden. Allerdings muss hierfür die Beschreibung der Software aus Prozess- oder Objektsicht erfolgen. Eine Beschreibung aus Aufbau- oder Aufgabensicht ist lediglich für die Überblicksdarstellung von

Aufgaben, Teilaufgaben und Organisationseinheiten sinnvoll. [Sch1998; Win2000]

Unter graphischen oder visuellen Sprachen werden formale Sprachen verstanden, deren Syntax, Semantik und Zeichensetzung auf Objekten basiert, welche nur über visuelle Kanäle wahrgenommen werden können [Sch1998].

Im Folgenden werden einige verbreitete graphische Softwarebeschreibungssprachen und Programmiersprachen vorgestellt, welche sich für die Modellierung von Software für mechatronische Systeme als geeignet erwiesen haben.

Funktionsbausteinsprache

Die Funktionsbausteinsprache besitzt die Form einer Datenflussdarstellung. In Abbildung 7 ist als Beispiel eine Funktionsbausteinkette gezeigt.

Hier werden zunächst jeweils zwei Zahlen miteinander addiert, um später miteinander verglichen zu werden. Wie auch in diesem Beispiel angedeutet, lassen sich mit solchen Datenflussdarstellungen sehr gut mathematische und logische Operationen modellieren. Zeitliche Abläufe sind mit einer solchen Sprache nur schwer zu realisieren, da alle in einem Datenflussplan enthaltenen Funktionsbausteinketten synchron getriggert werden. Ein Datenflussplan erlaubt daher auch die Darstellung einer parallelen Verarbeitung von Anweisungen, was jedoch nur Sinn macht, wenn der Mikroprozessor dies überhaupt unterstützt. [NGL2000]

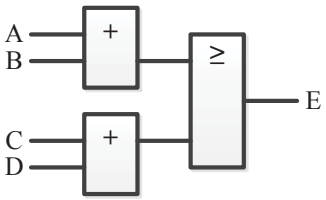

Abbildung 7: Beispielprogramm in Funktionsbausteinsprache [DIN61131-3-2003]

Kontaktplan

Genau wie die Funktionsbausteinsprache ist der Kontaktplan ebenfalls synchron getriggert und ermöglicht die Darstellung von Datenflüssen. Vorrangig wurde er jedoch für die Verarbeitung von binären Signalen konzipiert, weshalb er für komplexere Rechenoperationen nicht geeignet ist. In Abbildung 8 ist ein Beispiel eines Kontaktplans dargestellt. Dabei wird der Ausgang D auf logisch 1 gesetzt, wenn entweder A und B logisch 1 sind oder wenn C logisch 1 ist, ansonsten ist D logisch 0. [NGL2000]

Ablaufsprache

Reaktive Systeme können gut mit Hilfe von Sprachen des Zustandsübergangsparadigmas beschrieben werden [Win2000]. Die Ablaufsprache ist ereignisgesteuert. Das bedeutet, dass hier auf das Eintreten bestimmter Ereignisse reagiert wird. Dies kann beispielsweise der Fall sein, wenn ein bestimmter Zeitpunkt erreicht wird oder ein Nutzer eine bestimmte Aktion durchführt. Ein bekannter Vertreter ist die Ablaufsprache, bei welcher die Funktionalität der Software in Form eines Ablaufdiagramms dargestellt werden kann. Dieses besteht aus einzelnen Zuständen und gerichteten Verbindern. Während der Ausführung des Programms werden die Zustände und Verbinder durchlaufen. Dabei werden die vorgegebenen Bedingungen an den Verbindern berücksichtigt. Zeitliche und zustandsabhängige Abläufe können so dargestellt werden.

In Abbildung 9 ist ein Beispiel einer LED-Steuerung (LED = lichtemittierende Dioden) dargestellt, bei welcher die LED entweder automatisch ein- und nach 30 Sekunden wieder ausgeschaltet wird oder durch einen Tastendruck ein- und nach 20 Sekunden wieder ausgeschaltet wird.

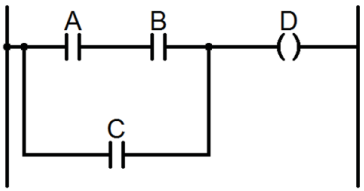

Abbildung 8: Beispiel eines Kontaktplans [DIN61131-3-2003]

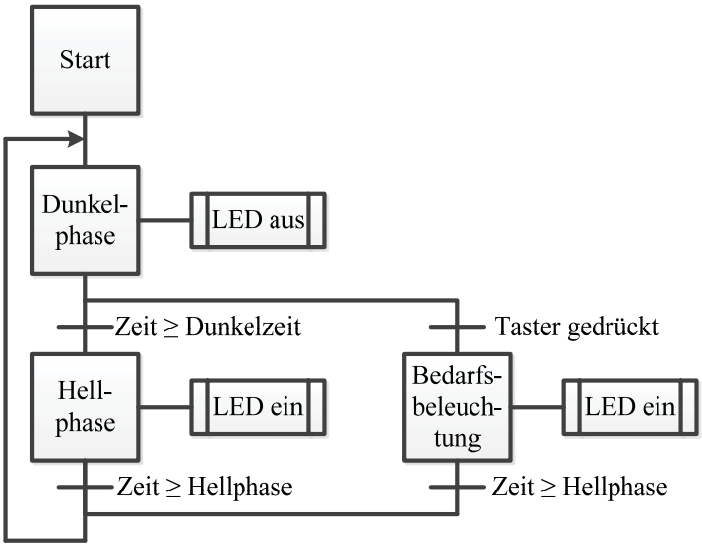

Abbildung 9: Beispielprogramm in einem Ablaufdiagramm als graphische Darstellung der Ablaufsprache [DIN61131-3-2003]

Im Ablaufdiagramm wird die logische und zeitliche Abfolge bestimmter Funktionen festgelegt. Die jeweiligen Funktionen, wie hier zum Beispiel „LED ein“ und „LED aus“, müssen an anderer Stelle definiert sein. Einzelne freie Berechnungen, wie bei der Funktionsbausteinsprache, sind nicht ohne weiteres möglich. [NGL2000]

Nassi-Shneiderman-Diagramme

Genau wie die Ablaufsprache sind Nassi-Shneiderman-Struktogramme [DIN66261-1985] ereignisgesteuert. Sie sind daher ebenso für reaktive Systeme geeignet.

In Abbildung 10 ist der aus Abbildung 9 bekannte Programmablauf noch einmal in Form eines Struktogramms nach Nassi-Shneiderman dargestellt.

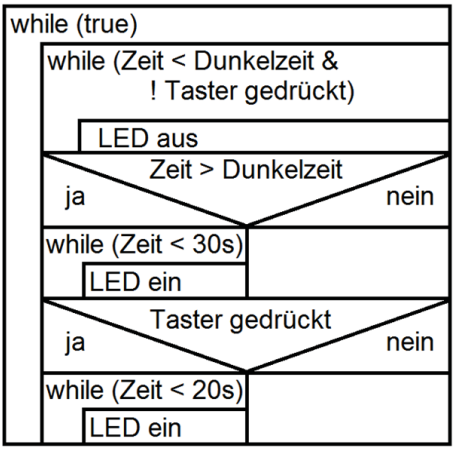

Abbildung 10: Struktogramm nach Nassi-Shneiderman [DIN66261-1985]

Unified Modeling Language

Die Unified Modeling Language (UML) [ISO/IEC19501-2005] wird hier nur der Vollständigkeit halber genannt, da sie für den Einsatz in Verbindung mit Mikrocontrollern nur bedingt geeignet ist. UML ist eine mächtige Beschreibungssprache für den Entwurf komplexer Softwarearchitekturen und beinhaltet viele unterschiedliche Werkzeuge, wodurch sie selbst sehr komplex wird und viele Dialekte besitzt. Aus diesem Grund ist das Erlernen von UML vergleichsweise schwierig, weshalb die Sprache fast ausschließlich von Experten für komplexe Entwicklungen genutzt wird. [RQS2012]

Die Werkzeuge der UML basieren auf objektorientierten Ansätzen [Oes2012]. Bei der Mikrocontrollerprogrammierung kommen diese Ansätze auch bei textuellen Sprachen gewöhnlich nicht zum Einsatz, da sie einen unverhältnismäßigen Overhead generieren, was bei der ohnehin schon geringen Verarbeitungsleistung von Mikrocontrollern unerwünscht ist [BST2010].

Weitere Sprachen

Natürlich existieren neben den aufgeführten noch weitere Sprachen. Allerdings handelt es sich bei diesen nur um Kombinationen und Abwandlungen.

So nutzt die in LabVIEW genutzte Sprache „G“ die datenflussorientierte Funktionsbausteinsprache, welche durch domänenspezifische Bestandteile erweitert wurde [Jun2000].

Die verbreitetsten Implementierungen der genannten Sprachen werden in Abschnitt 2.4 näher vorgestellt.

2.3 Grundlagen zur Mensch-Maschine-Schnittstelle

Im Rahmen dieser Arbeit sollen Untersuchungen an Entwicklungssystemen vorgenommen werden, welche durch Nutzer eingesetzt werden sollen. Dabei spielt die Programmierung der zu entwickelnden Systeme eine zentrale Rolle.

Graphische Programmier- und Softwarebeschreibungssprachen sowie Entwicklungsumgebungen, in welche diese integriert sind, können als Mensch-Maschine-Schnittstelle angesehen werden [Jun2000]. Einige Methoden der Evaluierung von Mensch-Maschine-Schnittstellen wurden im Rahmen dieser Arbeit für die Untersuchung bestehender Sprachen und für die Entwicklung und Implementierung neuer Sprachkonzepte genutzt. Daher wird an dieser Stelle ein kurzer Einblick in die wichtigsten Grundlagen auf diesem Gebiet gegeben.

2.3.1 Mensch-Maschine-Schnittstelle

Die Mensch-Maschine-Schnittstelle (oder auch Benutzungsoberfläche [VDI3699-2005]) wird in der DIN EN ISO 9241-110 definiert als „alle Bestandteile eines interaktiven Systems (Software oder Hardware), die Informationen und Steuerelemente zur Verfügung stellen, die für den Benutzer notwendig sind, um eine bestimmte Arbeitsaufgabe mit dem interaktiven System zu erledigen“ [DIN9241-110-2008]. Sie stellt damit in einem Mensch-Maschine-System das Bindeglied zwischen Mensch und Maschine her. In einem solchen System nutzt der Mensch die Maschine, um einen Prozess zu überwachen, zu steuern und zu regeln, wie dies in Abbildung 11 dargestellt ist [Joh1993; She1992].

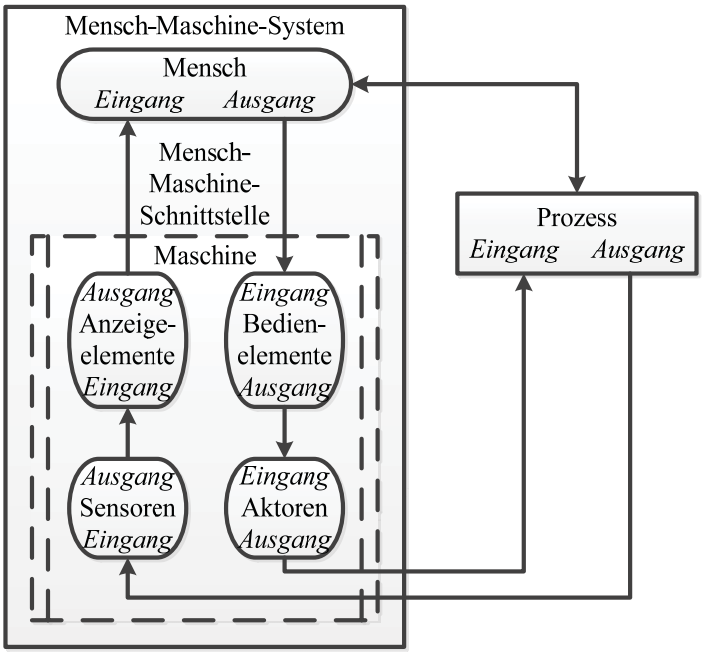

Abbildung 11: Interaktionen von Mensch mit Maschine und Prozess angelehnt an Sheridan [She1992] und Noyes/Baber [NB1999]

Die Zahl der Mensch-Maschine-Systeme wächst kontinuierlich an und es ist nicht abzusehen, dass dieser Trend abreißen wird. Die heute immer komplexer werdenden Arbeitsaufgaben und Prozesse erfordern die Unterstützung des Menschen durch Maschinen, allein schon um die Qualität der Produkte zu gewährleisten.

Um dem Menschen allerdings wirklich eine Unterstützung zu sein und keine zusätzliche Belastung, ist es notwendig die Schnittstelle zwischen Mensch und Maschine möglichst so zu gestalten, dass sie im Nutzungskontext gebrauchstauglich ist [DIN9241-100-2010].

Gebrauchstauglichkeit ist in der DIN ISO/TR 9241-100 definiert als „das Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem bestimm-

ten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen“ [DIN9241-100-2010].

2.3.2 Ergonomische Anforderungen

Um eine möglichst gute Gebrauchstauglichkeit zu ermöglichen, ist es wichtig, die Mensch-Maschine-Interaktion ergonomisch zu gestalten. So ist es möglich, Fehler zu vermeiden und die Arbeit mit einem interaktiven System zu erleichtern. In der Normenreihe DIN EN ISO 9241 werden verschiedene Empfehlungen für die Gestaltung interaktiver Systeme gegeben. Einen Überblick über die einzelnen Teile der Reihe bietet die DIN ISO/TR 9241-100. [DIN9241-100-2010]

Im Rahmen dieser Arbeit wird vorrangig die softwareseitige Ergonomie untersucht, für deren Betrachtung die DIN EN ISO 9241 allgemeine Richtlinien sowie Grundsätze der Dialoggestaltung und der Interaktionsgestaltung beinhaltet [DIN9241-100-2010].

Die sieben Grundsätze der ergonomischen Dialoggestaltung sind [DIN9241-110-2008]

- Aufgabenangemessenheit,
- Selbstbeschreibungsfähigkeit,
- Erwartungskonformität,
- Lernförderlichkeit,
- Steuerbarkeit,
- Fehlertoleranz und
- Individualisierbarkeit.

Diese Grundsätze bilden lediglich die Basis für die Beurteilung der Ergonomie eines Dialogs. Zur Bewertung selbst können sie nur schwer genutzt werden, da sie sehr allgemein gehalten sind. Daher werden in den weiteren Normen der ISO 9241-Reihe explizite Empfehlungen aufgeführt. Diese beschäftigen vorrangig mit [DIN9241-110-2008]:

- der Informationsdarstellung (ISO 9241-12),

- der Benutzerführung (ISO 9241-13) und
- der Dialogführung (ISO 9241-14 bis 9241-17).

Zur Beurteilung der Gestaltung von Interaktionen können die ISO 9241-120 bis ISO 9241-129 genutzt werden. Diese beinhalten die folgenden Leitlinien [DIN9241-100-2010]:

- Grundsätze der Darstellung
- Auswahl und Kombination von Medien
- Multimodale Interaktion und Navigation
- Bildliche Darstellung
- Akustische Interaktion (einschließlich Spracheingabe)
- Taktile/haptische Interaktion

Die Empfehlungen der Normen zeigen Standards auf, welche bei der Entwicklung eines interaktiven Systems helfen können. Sie müssen auf den jeweiligen Nutzungskontext angepasst werden und sind somit als Anwendungsrahmen zu verstehen, nicht als Beschreibung von Dialoganforderungen. [DIN9241-110-2008]

2.3.3 Der Nutzer im Zentrum der Entwicklung

Den Entwicklern ist heutzutage im Allgemeinen bekannt, dass der spätere Nutzer im Zentrum der Entwicklung eines Systems stehen sollte, so dass nicht nur die gewünschten Funktionen des Systems in die Anforderungsanalyse einer Entwicklung einfließen, sondern auch die Eigenschaften der Nutzer [NB1999; VIR2002]. Durch diese Erkenntnis hat sich die Qualität von Mensch-Maschine-Schnittstellen in den letzten Jahren deutlich gesteigert. War zum Beispiel vor einigen Jahren die Inbetriebnahme eines modernen Fernsehers ohne Studium der Betriebsanleitung nur schwer möglich, sind heute, trotz des gestiegenen Funktionsumfangs, wenigstens die grundlegenden Funktionen auch für Laien nutzbar.

Auch wenn dieser Ansatz offenbar schon sehr gute Fortschritte brachte, zeigt sich dennoch an verschiedenen Stellen, dass Systeme nicht intuitiv nutzbar sind. Ein Grund hierfür besteht darin, dass die Entwickler nicht sel-

ten gar keine Nutzer persönlich einbeziehen, sondern ihre Informationen (falls überhaupt) nur aus sekundären Quellen beziehen. Oft sind die Entwickler der Meinung, dass sie die späteren Nutzer ihres Systems ausreichend genau einschätzen können und ziehen aus dem eigenen Unternehmen Mitarbeiter aus der Vertriebsabteilung und eventuell noch Experten auf den Gebieten der Ergonomie und des Designs zu rate. Auch werden Tests der Software oft nur mit Mitarbeitern des eigenen Unternehmens durchgeführt. Diese Vorgehensweise wird oft aus Zeit- und Kostengründen gewählt und führt nicht unbedingt zum gewünschten Ergebnis. Daher ist es wichtig, eine signifikante Anzahl der späteren Nutzer in den Entwicklungsprozess einzubinden, um ein möglichst ergonomisches und nutzerfreundliches System entwickeln zu können. Dies bringt verschiedene Vorteile. Auf der einen Seite werden die Motivation und das Wohlbefinden auf Seiten des Nutzers durch eine angepasste Gestaltung gesteigert. Auf der anderen Seite werden so die Fehlerraten sinken, wodurch die Produktivität steigt. [Züh2004]

Um Fehlerraten durch eine angepasste Gestaltung zu reduzieren, muss die Komplexität der Schnittstelle auf die Arbeitsaufgabe und die Systemdynamik abgestimmt sein. So sind die Auswahl der dargestellten Informationen und die Form der Darstellung in einem komplexen, dynamischen System, wie beispielsweise einem Atomkraftwerk, sehr wichtig. Sowohl das Darstellen aller verfügbaren Daten als auch das Unterschlagen wichtiger Daten können zu Fehlentscheidungen führen. Zudem kann bei unangepasster Darstellung und hoher Systemkomplexität eine Zuordnung von Fehlern und deren Folgen erschwert werden [Mar2008]. [Dör1992]

2.3.4 Ziele der Usability-Evaluierung

Die ersten Maschinen, welche in Produktionsprozesse eingebunden waren, wurden noch nicht nach Gesichtspunkten der heutigen Gebrauchstauglichkeit entwickelt. Die Bedienung der Maschinen orientierte sich noch rein an deren Funktionalität und der Nutzer musste sich der Maschine unterordnen. Dies ist insofern ein interessanter Fakt, da es bereits viele gut auf den Menschen abgestimmte nichttechnische Produkte gab, wie zum Beispiel Bücher, welche bereits eine fast perfekte Schnittstelle zum Lesen bereitstellten [Coy2008]. Zu Beginn der Mensch-Maschine-Ära machten sich die Entwickler dennoch nur wenige Gedanken darüber, wie gut die Ma-

schinen bedienbar wären, da sich der flexible Mensch ja gut an die starren Maschinen anpassen könne. Zu dieser Zeit war das aber auch kein großes Problem, da die Funktionalität dieser Maschinen doch eher beschränkt war. Mit steigender Komplexität der Arbeitsaufgaben der Maschinen musste diese funktionsorientierte Entwicklungsstrategie allerdings überdacht werden, um gebrauchstaugliche Systeme entwickeln zu können. Heute geht man davon aus, dass Benutzungsschnittstellen von Maschinen so gestaltet sein müssen, dass diese dem Menschen die Aufgaben erleichtern und Ziele möglichst effizient erreicht werden können, indem die Mensch-Maschine-Schnittstelle möglichst, anforderungsgerecht und ergonomisch optimal aufgebaut ist [KRT2010].

Der Begriff der Gebrauchstauglichkeit, welcher im englischen Sprachraum als Usability bezeichnet wird, ist in der DIN EN ISO 9241-11 definiert [DIN9241-11-1999]. Diese und andere Normen stellen diverse allgemeine und spezielle Vorschriften zur Gestaltung von Maschinenarbeitsplätzen bereit, welche allesamt den Nutzer und die Arbeitsaufgabe ins Zentrum der Gestaltung stellen. Dennoch kann nicht gewährleistet werden, dass durch stures Einhalten der Vorgaben der Normen, die erforderliche Gebrauchstauglichkeit eines Produktes erreicht wird, da die Kombination der einzelnen Merkmale nicht zwangsläufig zu einem ausreichend guten Ergebnis führt [DIN9241-11-1999]. Um ein gutes Ergebnis zu erreichen, ist ein an den Zweck und Nutzungskontext des Systems angepasstes Vorgehen notwendig. Im Zentrum dieses Vorgehens steht zunächst eine Anforderungsanalyse, in welcher die Gebrauchstauglichkeitszielstellungen erarbeitet werden. Nach Mayhews' „Usability Engineering Lifecycle" müssen dazu die vier in Abbildung 12 dargestellten Punkte untersucht werden [May2008].

In der DIN EN ISO 9241 sind diese vier Punkte wiederzufinden. Auch hier sollen Aussagen zu den Benutzern (Nutzerprofil), zur Arbeitsaufgabe (Aufgabenanalyse), zu den Arbeitsmitteln (Plattformbesonderheiten/Einschränkungen) und zur Umgebung (grundlegende Designprinzipien) zusammengetragen werden und in den Entwicklungsprozess einfließen. Mit Hilfe dieser Informationen können die in der DIN EN ISO 9241 geforderten Ziele der Gebrauchstauglichkeit erreicht werden. [DIN9241-11-1999]

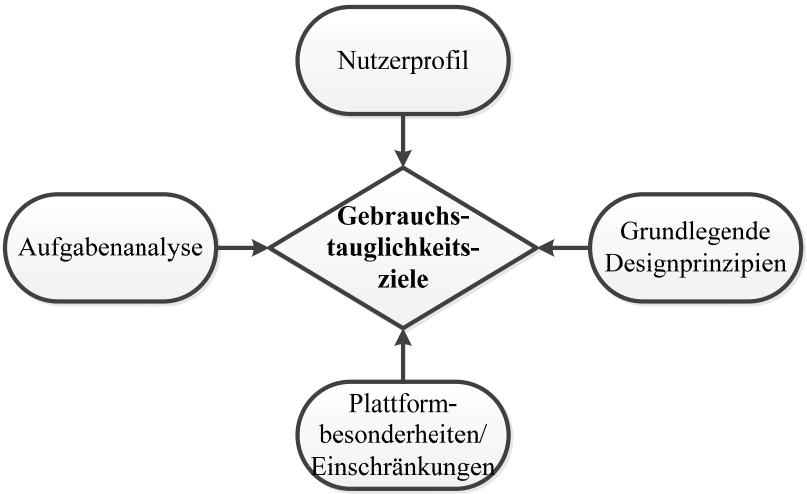

Abbildung 12: Einflüsse auf die Gebrauchstauglichkeitsziele
(engl. Usability Goals)

2.3.5 Methoden der Usability-Evaluierung

Um die in Abschnitt 2.3.4 angesprochenen Zielstellungen effizient, effektiv und zufriedenstellend erreichen zu können, wurde eine Vielzahl verschiedener Methoden zur Usability-Evaluierung entwickelt.

Abbildung 13 zeigt die wichtigsten Methoden nach Sarodnick und Brau unterteilt in empirische und analytische Methoden [SB2011]. Einteilungen nach anderen Kategorisierungskriterien sind auch möglich. Allerdings ist die strikte Aufteilung der Methoden in die Kategorien oft nicht möglich, da verschiedene Methoden oft auf den Nutzungskontext angepasst werden müssen und damit in eine andere Kategorie rutschen beziehungsweise in mehreren Kategorien erscheinen können. [Che2009]

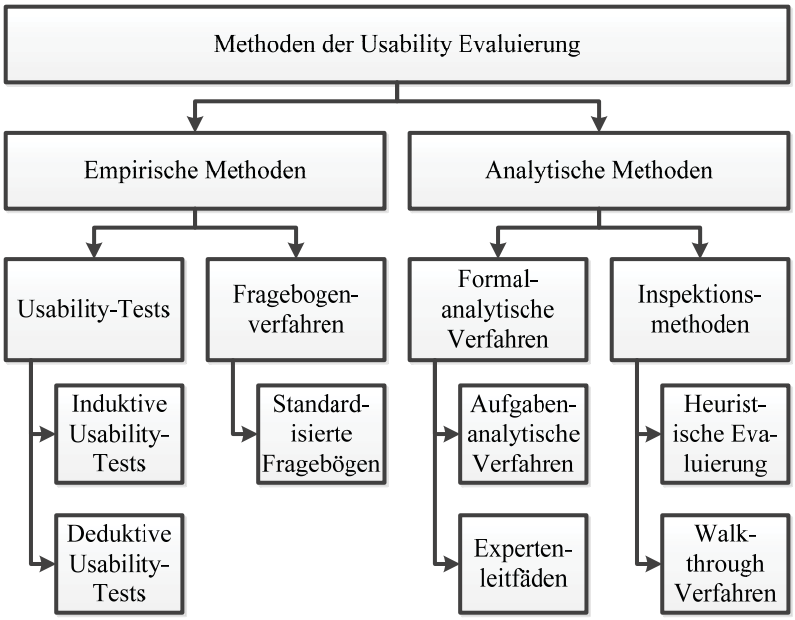

Abbildung 13: Methoden der Usability-Evaluierung

Während bei den empirischen (benutzerorientierten) Methoden die Bewertung der Usability vor allem anhand von Befragungen und Beobachtungen der Nutzer erfolgt, wird bei analytischen (expertenorientierten) Methoden beispielsweise die Einhaltung von Gestaltungsrichtlinien untersucht, wobei das Verhalten der Nutzer antizipiert wird [ST2003]. Bei der Anwendung analytischer Methoden kann meist auf die Nutzung von Prototypen verzichtet werden, da die Nutzer nicht am Evaluierungsprozess beteiligt sind. Bei den empirischen Methoden hingegen kann nur selten auf den Prototyp verzichtet werden, da sich die Nutzer meist keine genaue Vorstellung des Systems machen können. [SB2011]

Somit kann davon ausgegangen werden, dass analytische Methoden im Allgemeinen sinnvoll am Anfang eines Entwicklungsprozesses eingesetzt werden können, wohingegen empirische Methoden eher für die Evaluierung bestehender Systeme oder prototypischer Systeme einsetzbar sind. [SB2011]

Bei der Anwendung einiger der genannten Methoden wird die Mitwirkung von Usability-Experten empfohlen, da die jeweilige Erfahrung des Evaluierenden einen entscheidenden Einfluss auf das Ergebnis haben kann [Bia2011; BM2005]. Andere Methoden wurden mittlerweile gut formalisiert und sind auch für Laien nutzbar [Nie2005]. Im Rahmen der formal-analytischen Methoden ist dies bereits gelungen. So sind beispielsweise verschiedene Richtlinien und Fragebogenverfahren auch mit geringem Vorwissen einsetzbar.

Einige der in Abbildung 13 genannten Methoden werden in dieser Arbeit genutzt und daher im Folgenden kurz vorgestellt. Vorher werden einige hierfür wichtige Rahmenbedingungen diskutiert.

Nutzerbeteiligung

Wie bereits erwähnt, sind die späteren Nutzer während einer analytischen Evaluierung nicht direkt beteiligt. Dennoch ist es wichtig, dass die evaluierende Person ein genaues Bild von den Nutzern hat und somit die Handlungen realistisch voraussagen kann. Hier kann im Vorfeld der Evaluierung, ähnlich wie bei der empirischen Evaluierung, beispielsweise eine Beobachtung der Nutzer an vergleichbaren, existierenden Systemen erfolgen. Methoden wie das Thinking-Aloud, bei welchem die Nutzer während des Tests ihre Gedankengänge laut äußern, bieten sich hier an. Bei diesen Methoden muss allerdings eine Einschätzung der Zuverlässigkeit der gewonnenen Informationen erfolgen [Kie1997]. Als Ergebnis liegen am Ende eines solchen Vorgehens unterschiedliche Nutzermodelle für die Evaluierung vor.

Nutzermodelle

Ein Nutzermodell ist nicht, wie der Name dies vermuten lässt, eine Ansammlung von vereinfachtem und generalisiertem Wissen über mögliche Nutzer. Vielmehr beschreibt ein Nutzermodell, wie ein möglicher Nutzer bestimmte Bestandteile eines ihm vorgestellten Systems verstanden hat. Es ist daher ein mentales Modell eines technischen Systems, welches sich aus der Interaktion des Nutzers mit dem System ergibt. Es entwickelt sich aufgrund bestimmter mentaler Tätigkeiten, welche für jeden Menschen, abhängig von der jeweiligen Erfahrung, Wahrnehmung, Schlussfolgerung und anderer mentaler Prozesse, verschieden sind. Daher besitzen unterschiedliche Menschen gewöhnlich auch unterschiedliche Nutzermodelle des gleichen

Systems. Um möglichst gute Systeme zu entwickeln, ist es wichtig, dass Entwickler sich eine Vorstellung vom Nutzermodell einer bestimmten Nutzergruppe machen können. Hierfür müssen der Nutzer und seine Arbeitsaufgaben analysiert werden. Diese Analyse kann zum Teil zwar durch Recherchen erfolgen, jedoch ist die Erstellung eines vollständigen Nutzermodells nur möglich, wenn reale Nutzer einbezogen werden. [Züh2004]

Zentrale Fragestellungen im Rahmen der Usability Evaluierung

Nach Gediga und Hamborg geht man bei der Nutzung aller Usability-Evaluierungs-Methoden grundsätzlich von drei Fragestellungen aus [GH2002]:

- Zunächst wird im Rahmen eines Vergleichs mehrerer möglicher Systemeigenschaften hinterfragt, welche dieser Systemeigenschaften bestimmte Bewertungskriterien am besten erfüllen. Hierfür müssen natürlich mehrere Alternativen zur Verfügung stehen.
- Da diese relative Aussage nicht ausreichend ist, wird im zweiten Teil hinterfragt, wie gut die Bewertungskriterien erfüllt sind. Diese beiden Fragen bilden die Basis zur Einstufung und Auswahl der besten Alternativen. Allerdings ist bei der Betrachtung der Usability nicht nur die Untersuchung der Kombinationen der besten Alternativen wichtig. Es müssen genauso die Schwachstellen untersucht werden.
- Die dritte Frage soll daher klären, warum bestimmte Systemeigenschaften anhand der Bewertungskriterien nicht als positiv bewertet werden konnten.

Die Bewertungskriterien können dabei aus verschiedenen Quellen stammen. Oft werden ISO-Normen oder Design-Richtlinien genutzt, welche allerdings zunächst an das entsprechende Problem angepasst werden müssen. Hierfür lassen die Normen und Richtlinien Spielräume offen, da sie eher einen Rahmen vorgeben sollen als strikte Regeln.

Usability-Tests

Im Rahmen dieser Arbeit werden einige Ergebnisse von Usability Tests vorgestellt. Daher werden der prinzipielle Ablauf sowie eine wichtige und in der Arbeit genutzte Technik dieser empirischen Methode hier vorgestellt.

Der simple Grundgedanke hinter den Usability-Tests ist, dass Informationen direkt von Nutzern gesammelt werden, welche das Produkt einmal getestet haben. Dies ist in Form von Beobachtungen und Messungen während der Nutzung des Produkts oder in Form von anschließenden Interviews möglich. Offenbar scheint dieses Vorgehen bei den meisten Menschen als sinnvoll angesehen zu werden, weshalb diese Methode zu den bekanntesten und am meisten genutzten gehört. Wie viele andere Methoden entwickelte sich auch diese im Verlauf der Zeit. Basierten die Aussagen anfangs eher auf harten, objektiven Fakten, wie Zeit- und Fehlermaßen, so wurde später eher eine weichere, verhaltensbasierte Herangehensweise angewendet. [SB2011]

Unter welchen Rahmenbedingungen die Tests durchgeführt werden, hängt von verschiedenen Faktoren ab. Beispielsweise kann es für einige Tests sinnvoll sein, sie im gewöhnlichen Arbeitsumfeld durchzuführen, um hier bestehende Einflüsse, wie Lichteinstrahlung oder klimatische Belastungen, bewerten zu können. Andere Tests hingegen sollen unabhängig von im Arbeitsumfeld bestehenden Einflüssen, wie z.B. Ablenkungen durch Kollegen, durchgeführt werden. Dies ist einzelfallabhängig.

Unabhängig vom gewählten Ort bearbeiten die Nutzer immer realistische oder sogar reale Aufgaben mit Hilfe des Produktes. Dabei werden sie von den Evaluatoren beobachtet. Sind die Aufgaben abgearbeitet, werden Befragungen in Form von Interviews oder Fragebögen zu ausgewählten Problemen durchgeführt und Schwierigkeiten bei der Arbeit besprochen. Auch Hintergrundinformationen über den Nutzer, wie seine Vorkenntnisse, werden an dieser Stelle erhoben.

Es zeigt sich, dass Usability-Tests sehr subjektiv und von den Probanden abhängig sind. Eine Auswahl möglichst repräsentativer Nutzer ist daher wichtig, um ein aussagekräftiges Ergebnis zu erhalten. Ist diese Auswahl gelungen, reichen oft schon vier bis sechs Personen aus, um ein zuverlässiges Ergebnis zu erhalten [KR1997; Nie1994].

Im Bereich der Usability-Tests haben sich aufgrund der großen Verbreitung sehr viele Einzelmethoden entwickelt, welche hier nicht im Einzelnen behandelt werden. Allerdings lässt sich eine Unterscheidung nach deduktiven und induktiven Methoden vornehmen. Induktive Tests untersuchen vorrangig Schwachstellen beziehungsweise Gestaltungs- und Verbesserungsmöglichkeiten. Deduktive Tests treffen eine Aussage über die Leistungsfähigkeit eines Produktes. [RSS1994]

Thinking-Aloud

Eine bei Usability-Tests oft angewendete und beliebte Technik ist das Thinking-Aloud. Dabei löst der Nutzer mit Hilfe des Produktes eine oder mehrere Aufgaben. Währenddessen versucht er seine Gedankengänge zu artikulieren, indem er seine Handlungen und die Beweggründe für diese beschreibt.

Mit dieser Technik erhalten Evaluatoren und Entwickler Informationen über die Gedankenabläufe des Nutzers, während er das Produkt nutzt. Durch das laute Mitdenken werden Probleme und Schwachstellen effektiver erkannt als bei der reinen Betrachtung der Handlungen. Beispielsweise kann erkannt werden, warum Nutzer zur Lösung von Teilproblemen nicht den optimalen Handlungsablauf nutzen, sondern einen längeren und für sie nachvollziehbareren Weg. [DF2009]

Bei der Evaluierung muss allerdings beachtet werden, dass die geäußerten Gedankengänge nur ein Teil der Technik sind. Sie dürfen nicht unabhängig von den Handlungen betrachtet werden, da es sehr oft einen Unterschied zwischen dem gibt, was die Nutzer denken und was sie machen. [Kie1997]

Kognitiver Walkthrough

Auch der kognitive Walkthrough wurde im Rahmen der Arbeit genutzt, weshalb er hier kurz erklärt wird. Er stellt unter den analytischen Methoden der Usability-Evaluierung ein häufig genutztes Verfahren dar. Warum dieses Verfahren so beliebt ist, hat vor allem zwei Gründe. Zum einen kann es sehr früh in der Entwicklungsphase angewendet werden, da nicht unbedingt ein fertiger Prototyp benötigt wird. Zum anderen ist die Arbeitsweise sehr stark an das explorative Vorgehen beim Erlernen einer neuen Software angelehnt, welches bei vielen Nutzern beobachtet werden kann. Dabei beschäftigt sich der Nutzer im Allgemeinen nicht mit Bedienungsanleitungen oder Lehrver-

anstaltungen, sondern er entdeckt das Programm, indem er die für seine Arbeit wichtigen Tätigkeiten erlernt [Pat2000].

Beim kognitiven Walkthrough wird zunächst ermittelt, welche Handlungsabläufe das Programm vorgibt, um die gestellten Aufgaben zu erfüllen. Anhand dieser Abläufe wird überprüft, ob die Nutzerschnittstelle so gestaltet ist, dass ein Nutzer diese Handlungen durchführen würde. Weiterführende Annahmen über den Nutzer werden nicht getätigt. Während der Anwendung der Methode sollen nicht nur Probleme der Schnittstelle, sondern auch Gründe für die Probleme ermittelt werden. Dabei konzentriert sich die analysierende Person weniger auf die Schnittstelle an sich, sondern eher auf die wahrscheinlichen Gedankengänge der theoretischen Nutzer, weshalb diverse Informationen über das Hintergrundwissen der Nutzer benötigt werden. [LW1997; WRL1994]

2.4 Existierende Entwicklungssysteme für mechatronische Systeme

In diesem Abschnitt werden einige erfolgreich genutzte Konzepte zur Entwicklung mikrocontrollerbasierter mechatronischer Systeme vorgestellt. Diese Konzepte reichen von den Standardentwicklungsumgebungen und Evaluierungsboards der jeweiligen Mikrocontrollerhersteller über graphische, plattformunabhängige Umgebungen bis hin zu modularen Robotiksystemen von Herstellern wie LEGO und fischertechnik. Bei der Untersuchung dieser Systeme wird vor allem auf Aspekte eingegangen, welche die Entwicklung des Systems für den Nutzer vereinfachen sollen. Die Vereinfachung kann auf der Software- und der Hardwareebene bestehen.

Da die meisten Hersteller auf ähnliche Konzepte für die Vereinfachung der Entwicklung zurückgreifen, werden nur einige beispielhafte Systeme vorgestellt. Hierfür werden diese in entsprechende Kategorien unterteilt.

2.4.1 Herstellerspezifische textuelle Entwicklungsumgebungen

Um einen Mikrocontroller programmieren zu können, muss vom Entwickler zunächst der entsprechende Quellcode geschrieben werden. Hierfür steht eine Vielzahl von Textbearbeitungsprogrammen zur Verfügung. Der Quellcode muss anschließend soweit übersetzt werden, dass er für den Mikrocontroller lesbar ist. Des Weiteren muss das so entstandene Programm auf den Mikrocontroller geladen werden. Die für diesen Vorgang benötigten Werkzeuge bezeichnet man als Toolchain. Sie ist für jeden Mikrocontrollertyp anders und muss vom Nutzer eingerichtet und parametriert werden.

Um diesen Schritt zu erleichtern, stellen die meisten Mikrocontrollerhersteller ihren Kunden eine selbstentwickelte Programmierumgebung zur Verfügung. Hierzu gehören beispielsweise

- Code Composer Studio [Tex2012] für Mikrocontroller von Texas Instruments,
- Atmel Studio [Atm2012] für Atmel ARM® Cortex™-M und Atmel AVR® Mikrocontroller,
- MPLAB® [Mic2011a] für Microchip Mikrocontroller,
- C-Control IDE [Con2010] für C-Control-Plattformen und
- Arduino Development Environment [Ard2012b] für Arduino Plattformen.

Um ein Mikrocontrollerprogramm mit Hilfe einer solchen Entwicklungsumgebung zu erstellen, muss vom Nutzer als erstes ein Projekt angelegt werden. Dabei wählt er den gewünschten Mikrocontroller aus und legt fest, mit welchem Programmieradapter dieser programmiert werden soll. Des Weiteren muss er angeben, in welcher Sprache der Quellcode entwickelt wird und welche Werkzeuge in der Toolchain Verwendung finden. Der Nutzer wird an dieser Stelle gewöhnlich unterstützt, indem ihm beispielsweise nur Werkzeuge vorgeschlagen werden, welche mit dem ausgewählten Mikrocontroller kompatibel sind, oder auch indem die Auswahl bestimmter Werkzeuge automatisch erfolgt.

Sind diese Einstellungen vorgenommen, muss der Nutzer den Quellcode erstellen und in das Projekt integrieren. Um die Übersetzung des Quellcodes zu starten und auf den Mikrocontroller zu laden, genügt gewöhnlich ein einfacher Tastendruck oder das Auswählen eines Unterpunktes in einem Menü. Der Rest wird von der Entwicklungsumgebung selbstständig ausgeführt. Dies gilt natürlich nur unter der Voraussetzung, dass die Hardware korrekt angeschlossen und die Software richtig parametriert ist.

Kritische Betrachtung

Die herstellerspezifischen Plattformen werden im Hobbybereich sehr häufig genutzt, da nur wenige kostenlose Alternativen zur Verfügung stehen, bei welchen die Erstkonfiguration nach der Installation vergleichsweise einfach durchzuführen ist.

Vorteile:

- Das Übersetzen des Quellcodes und das Laden auf den Mikrocontroller werden durch die Oberflächen deutlich vereinfacht. Der Nutzer muss sich dabei nicht um die Ausführung der einzelnen Werkzeuge kümmern.
- Die herstellerspezifische Entwicklungsumgebung unterstützt den Nutzer bei der Basiskonfiguration. Hierbei werden verschiedene Funktionen des Mikrocontrollers über die Menüs der Oberfläche aktiviert oder deaktiviert, was ansonsten nur umständlich über die Parametrierung der Werkzeuge oder über den Quelltext möglich wäre.
- Eine erste funktionsfähige Standardkonfiguration ist bereits vorausgewählt. Dies erleichtert die erste Inbetriebnahme.
- Die meisten Hersteller bieten parallel zu den Entwicklungsumgebungen auch Evaluierungsplattformen an. Dies sind Elektronikplatinen, auf welchen sich neben dem Mikrocontroller sämtliche zur Programmierung und zum Betrieb und zum Test benötigten Schaltungen befinden [BHK2009]. Zum Teil wird hierfür auch weitere Elektronik, wie Ein- und Ausgabegeräte, angeboten, so dass dem Nutzer einige Schritte der Elektronikentwicklung abgenommen werden.

- Im Vergleich zu vielen graphischen Programmiersprachen lässt die textuelle Programmierung dem Nutzer viele Freiheiten, so dass er die Software sehr flexibel an die Problemstellung anpassen kann.

Nachteile:

- Um das mechatronische System in Betrieb nehmen zu können, muss der Nutzer selbstständig Sensoren und Aktoren an den Mikrocontroller anschließen. Hierbei unterstützen die Entwicklungsumgebungen nicht, so dass Expertenwissen auf dem Gebiet der Elektrotechnik benötigt wird.
- Die Wandlung von Spannungssignalen in digitale Repräsentationen durch den Mikrocontroller muss vom Nutzer verstanden worden sein, so dass auch hier Expertenwissen benötigt wird. Auch die hierfür benötigten Datentypen müssen dem Nutzer bekannt sein.
- Die textuelle Programmierung ist für unerfahrene Nutzer eine erhebliche Hürde beim Einstieg in die Entwicklung mechatronischer Systeme, da textuelle Programmiersprachen gewöhnlich deutlich schwerer zu erlernen sind als graphische. Lediglich in der Arduino Development Environment wird versucht gegenzusteuern, indem die verwendete textuelle Sprache vereinfacht wird. Dennoch ist auch sie nicht intuitiv nutzbar.

2.4.2 Graphische plattformunabhängige Entwicklung

Um Mikrocontroller plattformunabhängig programmieren zu können, stehen nur wenige Entwicklungsumgebungen zur Verfügung. Am weitesten verbreitet sind dabei MATLAB Simulink [Mat2011a] und LabVIEW [Nat2012]. Diese werden im Folgenden vorgestellt.

Beide Umgebungen stellen dem Nutzer graphische Funktionsbausteine zur Verfügung, mit welchen physikalische Modelle erzeugt und simuliert werden können. Für diese Modelle werden Regelalgorithmen entwickelt und innerhalb der Simulation getestet. Die Entwicklung des Regelalgorithmus kann in vielen Fällen über rein graphische Methoden erfolgen. Nur in einigen Fällen muss zusätzlich textueller Quellcode hinzugefügt werden. Zeigen sich positive Ergebnisse während der Simulation, so kann der erzeugte Reg-

ler mittels eines zusätzlichen Werkzeugs beispielsweise in C-Code übersetzt und auf einen Mikrocontroller übertragen werden.

MATLAB Simulink

Der zentrale Aufgabenschwerpunkt von MATLAB Simulink ist die vergleichsweise einfache Entwicklung und numerische Simulation mathematischer Modelle, welche das Verhalten realer Systeme beschreiben [Mat2011a]. In Simulink werden hierfür Datenflusspläne mit Hilfe einzelner Funktionsbausteine erzeugt. Simulink stellt bereits eine große Auswahl von Funktionsbausteinen standardmäßig zur Verfügung. Dennoch können diese oft nicht alle benötigten Funktionalitäten abbilden, was in vielen Fällen darauf beruht, dass zeitliche Abläufe mit Hilfe eines Datenflussplans nur schwer darstellbar sind. Für solche Fälle kann sich der Nutzer eigene Funktionsbausteine definieren, welche er mit Hilfe von MATLAB-Code programmiert [Mat2012c].

Mit Hilfe zusätzlicher Toolboxen kann aus dem so erzeugten Datenflussplan Quellcode für die Zielplattform generiert werden. Hierfür muss MATLAB Simulink den entsprechenden Mikrocontroller unterstützen. Den Quellcode kann der Nutzer anschließend in eine mikrocontrollerspezifische Entwicklungsumgebung laden, wie beispielsweise das unter 2.4.1 genannte Code Composer Studio, Atmel Studio oder MPLAB®. Von hier aus kann der Code übersetzt und auf die Zielplattform übertragen werden. Dieses Vorgehen bietet den Vorteil, dass der Quellcode noch einmal vom Nutzer überprüft und gegebenenfalls optimiert werden kann.

Für die meisten unterstützten Mikrocontroller ist es mittlerweile möglich, den Quellcode mit Hilfe des Real-Time Workshops direkt aus MATLAB Simulink zu übersetzen und auf den Controller zu laden. Jedoch sind für viele Plattformen zahlreiche, umständliche Anpassungen notwendig, welche selbst erfahrene Nutzer regelmäßig vor Probleme stellen. Vor allem bei der Integration der Toolchain kann es zu Schwierigkeiten kommen.

LabVIEW

LabVIEW wurde in erster Linie zur virtuellen Darstellung messtechnischer Geräte entwickelt. Dies spiegelt sich auch im vollständigen Namen „Laboratory Virtual Instrument Engineering Workbench“ und in der Art der Pro-

grammierung wider. Um ein Programm in LabVIEW zu entwickeln, wird in zwei Fenstern parallel gearbeitet. Während im ersten Fenster ein Messinstrument Bauteil für Bauteil virtuell aufgebaut wird, erscheint im zweiten Fenster, dem „Blockdiagramm"-Fenster, für jedes dieser Bauteile ein „Block", welcher die Funktionalität des Bauteils darstellt. [GM2012]

Im Blockdiagramm findet die eigentliche Programmierung statt. Ähnlich wie bei MATLAB Simulink erfolgt diese in einem Datenflussplan. Auch hier werden einzelne „Blöcke" angeordnet und über Datenflüsse miteinander „verdrahtet". Jedoch stehen dem Nutzer zusätzlich Schleifen, Fallunterscheidungen und Sequenzen als Kontrollstrukturen zur Verfügung. Mit Hilfe der Schleifen können Gruppen von Blöcken im Datenflussplan mehrfach wiederholt werden. Durch die Fallunterscheidungen wird während der Laufzeit des Programms entschieden, welche Bestandteile des Graphs ausgeführt werden sollen und welche nicht. Über die Nutzung von Sequenzen gibt der Nutzer vor, in welcher Reihenfolge die Berechnungen des Datenflussplans ablaufen. [Müt2009]

In Abbildung 14 und Abbildung 15 wird beispielhaft der Aufbau eines Messinstruments mit dem zugehörigen Blockdiagramm gezeigt.

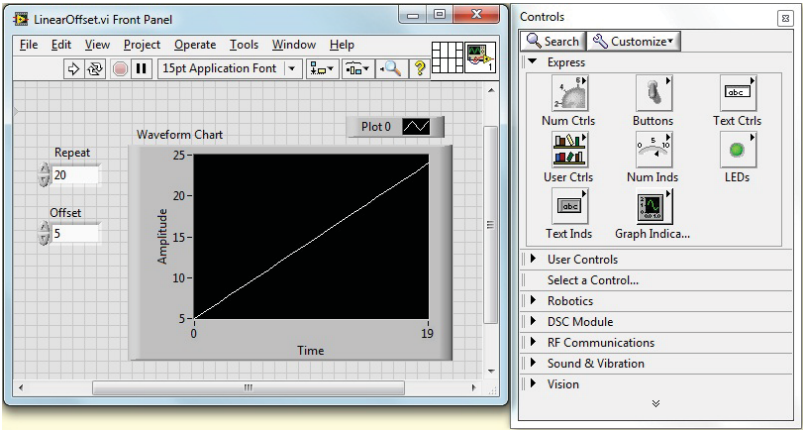

Abbildung 14: Oberfläche zum Aufbau eines virtuellen Messinstruments in LabVIEW

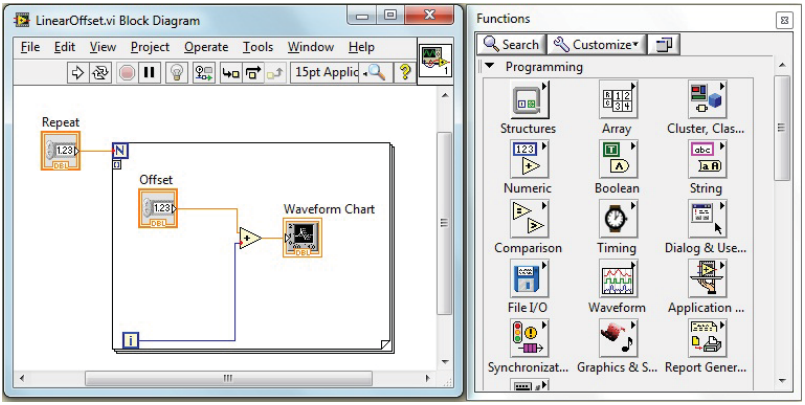

Abbildung 15: Oberfläche zum Erstellen des Blockdiagramms, welches die Funktion des Messgerätes definiert

Das Programm durchläuft dabei eine vorgegebene Anzahl Zeitschritte, was im vorliegenden Fall 20 sind. Bei jedem Durchlauf wird der Wert des aktuellen Zeitschrittes zu einem vom Nutzer vorgegebenen Offset hinzu addiert. Das Ergebnis wird in Form eines kontinuierlichen Graphen ausgegeben.

Um das Programm zu erzeugen, werden zunächst die beiden Bedienelemente und der kontinuierliche Graph in das erste Fenster eingefügt (Abbildung 14). Im Blockdiagrammfenster werden von LabVIEW automatisch die zugehörigen Blöcke hinzugefügt. Anschließend muss der Nutzer nur noch die Schleife und die Addition im Blockdiagramm platzieren und die benötigten Datenflüsse verbinden (Abbildung 15).

Um auch reale Messwerte erfassen und zur Ansteuerung von Aktoren nutzen zu können, muss eine entsprechende Hardware am Computer angeschlossen sein. Eine Möglichkeit hierfür ist die Nutzung von Datenerfassungssystemen, welche von LabVIEW unterstützt werden. Hier können die jeweiligen Mess- und Steuerkanäle direkt angesprochen werden. Wird ein Datenerfassungssystem nicht unterstützt, so kann dieses dennoch über eine Kommunikationsstrecke mit dem PC verbunden werden. Oft werden hierfür Signalprozessoren (DSP), FPGAs oder Mikrocontroller eingesetzt. Für eine ausgewählte Gruppe dieser Systeme kann LabVIEW aus dem Blockdiagramm Quellcode erzeugen, diesen kompilieren und auf die entsprechende Platt-

form laden. Dadurch ist es beispielweise möglich, eine Datenvorverarbeitung auf der Plattform durchzuführen und anschließend die Daten über eine Kommunikationsschnittstelle auf den PC an LabVIEW zu übertragen. [SS2007]

Kritische Betrachtung

Beide Produkte wurden nicht primär für die Programmierung von Mikrocontrollern entwickelt. Betrachtet man die jeweiligen Aufgabenspektren, für welche sie konzipiert wurden, so stellen sie fachkundigen Nutzern mächtige Werkzeuge auf unterschiedlichen Ebenen zur Verfügung. Für die Anwendung durch unerfahrene Nutzer sind sie jedoch nicht gedacht [Fra2010].

Vorteile:

- Es stehen verschiedene Plattformen für die Datenerfassung und -manipulation zur Verfügung, mit welchen zahlreiche mess- und regelungstechnische Aufgaben gelöst werden können.
- Die Plattformen bieten nach außen hin die üblichen Schnittstellen, so dass nur geringe Einschränkungen bei der Nutzung von Sensoren und Aktoren vorliegen.
- Die graphische Programmierung mit Hilfe von Datenflussplänen ist intuitiv und ermöglicht auch Nutzern ohne Programmierkenntnisse die Erzeugung von lauffähigem Quellcode.

Nachteile:

- Die Hardware außerhalb der genutzten Plattform wird im Rahmen der Entwicklung mechatronischer oder eingebetteter Systeme nicht betrachtet. Daher muss der Nutzer Zugriff auf Expertenwissen aus dem Bereich der Elektronik haben, um Sensoren und Aktoren mit der Datenerfassungs- und Datenmanipulationsplattform so verbinden zu können, dass die Funktionsfähigkeit gewahrt bleibt.

- Auch im Bereich der Signalwandlung wird davon ausgegangen, dass der Nutzer bereits die Zusammenhänge zwischen den gemessenen physikalischen Größen, den elektrischen Signalen und der jeweiligen Repräsentation im Mikrocontroller kennt. Hierzu gehört unter anderem, dass er die Existenz und die Eigenschaften unterschiedlicher Datentypen innerhalb des Mikrocontrollers verstanden hat.
- Es werden normierte Symbole und Bezeichnungen genutzt, was zwar die Nutzung für den Experten vereinfacht, jedoch für den unerfahrenen Nutzer zunächst unverständlich ist.
- Im Vergleich zur traditionellen C-Programmierung entsteht bei der Generation des Quellcodes ein deutlich erhöhter Bedarf an Speicher und Ausführungszeit. Aus diesem Grund beginnt beispielsweise bei LabVIEW die Welt der Mikrocontroller erst beim ARM-Cortex-M3 [Arm2006], einer leistungsfähigen 32-bit-Plattform. Mit weniger leistungsfähigen Systemen stößt der Nutzer schnell an unüberwindbare Grenzen. [SA2011]
- Im Hobbybereich sind die hohen Anschaffungskosten ein Problem. Diese resultieren vor allem aus dem hohen Funktionsumfang, welcher hier gewöhnlich nicht benötigt wird.

2.4.3 Modulare mechatronische Hardware mit graphischer Ablaufprogrammierung

Ein verbreitetes Konzept zur Vereinfachung der Entwicklung eines mikrocontrollerbasierten Systems ist die Bereitstellung von mechatronischen Hardwaremodulen, welche vom Nutzer mit Hilfe eines vereinfacht dargestellten graphischen Ablaufdiagramms programmiert werden. Hier sind verschiedene Hersteller zu nennen:

- fischertechnik GmbH mit der „ROBO“-Serie [Con2013]
- LEGO mit Roberta – Lernen mit Robotern und LEGO MINDSTORMS NXT [Leg2011a] und WeDo [Leg2011b]

Die Entwicklungsansätze dieser Systeme basieren auf vergleichbaren Prinzipien. Da LEGOs Roberta mit LEGO MINDSTORMS NXT eine Vorreiterrolle in diesem Bereich einnimmt, wird es in diesem Abschnitt stellvertretend vorgestellt. Die genutzten Entwicklungsprinzipien werden hieran aufgezeigt.

Roberta und LEGO MINDSTORMS NXT

Hinter dem Namen Roberta verbirgt sich ein Lehr- und Lernsystem, also ein technisches System, welches zur Vermittlung von Wissen eingesetzt werden kann [Kös2005]. Dieses soll bei Schülern Interesse an Technik, Wissenschaft und Informatik wecken und Wissen aus dem Bereich der Robotik sowie den zugehörigen Bereichen der Elektrotechnik, Physik und Mechanik veranschaulichen [BS2010]. Während im traditionellen Unterricht das Vermitteln von theoretischem Wissen vom Lehrer an den Schüler im Mittelpunkt steht [Mel1999], soll mit Hilfe moderner Lehr- und Lernsysteme eher das Verstehen technischen Wissens anhand praktischer Beispiele gefördert werden. Somit können Schüler motiviert werden, sich weiter mit technischen Themen auseinanderzusetzen. [Fra2009]

Die Hardware des Systems besteht aus einem Sortiment von Bausteinen zum Aufbau der Mechanik, einem zentralen elektronischen NXT-Baustein, Aktoren und verschiedenen Sensoren. Mit Hilfe dieser Hardware kann innerhalb kürzester Zeit ein Roboter in unterschiedlichen Formen aufgebaut werden. Sowohl die mechanischen Bausteine als auch die Sensoren und Aktoren können sehr einfach miteinander verbunden werden. Abbildung 16 zeigt den zentralen Baustein zusammen mit drei Motoren und vier Sensoren. Dies entspricht der jeweils maximal möglichen Anzahl. [Leg2011c]

Die Sensoren und Aktoren sind entweder über einen I²C-Bus oder über eine analoge Schnittstelle an den zentralen Baustein angeschlossen. Es existiert bereits eine große Auswahl unterschiedlicher Sensoren, allerdings ist es auch möglich, eigene Sensorik zu entwickeln, indem beispielsweise das NXT SuperPro Prototype Board genutzt wird [Hit2011]. Dabei übernimmt das Board die Funktion eines Wandlers, welcher ein analoges oder digitales Sensorsignal so wandelt, dass es im I²C-Bus zur Verfügung steht. Der Entwickler des Sensors bekommt hier eine fertige Plattform zur Verfügung gestellt, welche er nur noch programmieren muss. [Hit2011]

Abbildung 16: Zentraler NXT-Baustein des Roberta-Systems [Leg2011c]

Der zentrale NXT-Baustein des Roberta-Systems kann auf verschiedene Arten programmiert werden. Der Einstieg in die Nutzung des Systems erfolgt gewöhnlich mit Hilfe der intuitiv nutzbaren Programmierumgebung LEGO MINDSTORMS Education / Retail NXT, welches die graphische Programmiersprache NXT-G nutzt. Abbildung 17 zeigt die Programmierumgebung. Diese muss lediglich auf dem PC installiert werden und ist ohne weitere Anpassungen einsatzbereit. In der Programmierumgebung wird die gewünschte Funktionalität erzeugt, indem einzelne Programmierblöcke in Form eines vereinfacht dargestellten Ablaufdiagramms zusammengesetzt werden. Die einzelnen Programmierblöcke können entsprechend parametriert werden, um zum Beispiel die Drehzahl der Motoren zu erhöhen. [Leg2011c]

Die Toolchain zur Programmierung des Mikrocontrollers ist in der Programmierumgebung hinterlegt, so dass sich der Nutzer keine Gedanken um diese Abläufe machen muss. [Leg2011c]

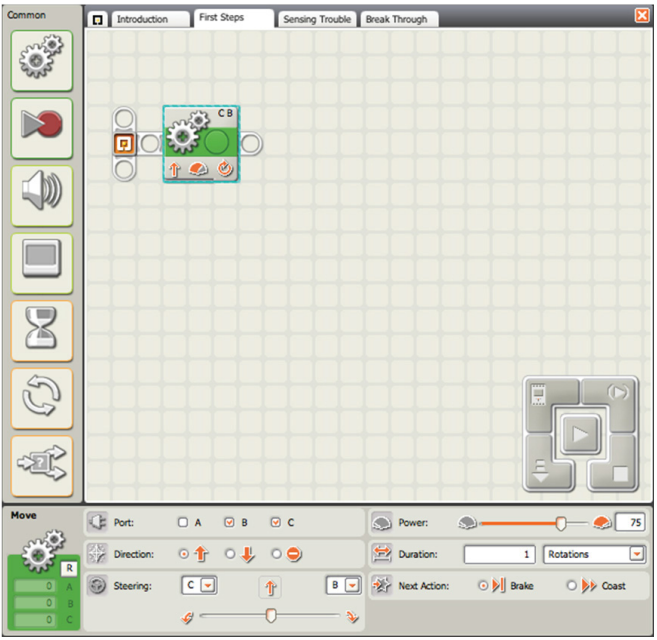

Abbildung 17: LEGO MINDSTORMS NXT Programmierumgebung [Leg2011c]

Weitere Möglichkeiten der graphischen Programmierung

Zwei weitere Möglichkeiten der graphischen Programmierung werden durch das bereits in Abschnitt 2.4.2 vorgestellte LabVIEW und RoboLab zur Verfügung gestellt. Allerdings wird LabVIEW, wie bereits beschrieben, nur für fortgeschrittene Nutzer empfohlen [Fra2010]. RoboLab basiert zwar auf LabVIEW [Nat2011], ist jedoch einfacher nutzbar als dieses und wird zum Teil auch in Schulen eingesetzt. Dies resultiert aber hauptsächlich daraus, dass die Roberta-Systeme anfangs mit dieser Software ausgeliefert wurden [Fra2007].

Das Fraunhofer Institut IAIS merkt in seinem Bericht zum Roberta-Projekt jedoch an, dass RoboLab ingenieursmäßiges Denken voraussetzt [Fra2007]. Daher ist es eher nur für erfahrene LEGO MINDSTORMS Anwender geeignet [Lpe2006].

Die Programmierung mit RoboLab erfolgt ähnlich wie mit der Programmierumgebung LEGO MINDSTORMS Education / Retail NXT. Auch hier müssen Ablaufpläne aus einzelnen Programmierblöcken zusammengesetzt und zum Teil parametriert werden. Gerade diese Parametrierung kann sehr frei gestaltet werden. Aber auch ansonsten besitzt RoboLab einen größeren Funktionsumfang als die eigentliche NXT-Programmierumgebung [Lpe2006]. Auch bei der Programmierung mit RoboLab muss sich der Nutzer nicht um die Toolchain zum Übersetzen und Brennen der entwickelten Software auf den Mikrocontroller kümmern, da diese in der Entwicklungsumgebung hinterlegt ist.

Möglichkeiten der textuellen Programmierung

Es existieren eine Reihe weiterer Programmiermöglichkeiten mittels textueller Programmiersprachen. Die am meisten genutzten Sprachen sind NXC, leJOS, RobotC, GNAT GPL for NXT und PBLua. NXC und RobotC sind vergleichbar mit C, leJOS ist ein Java-Derivat, GNAT GPL for NXT basiert auf Basic und PBLua auf Lua. Diese Sprachen sind jedoch nur für fortgeschrittene Anwender zu empfehlen, nicht für Neueinsteiger. [Fra2010]

Kritische Betrachtung

Bei den vorgestellten Systemen wird nicht auf der Ebene des Mikrocontrollers gearbeitet, sondern auf einer höheren Ebene, welche nur die mechanischen beziehungsweise mechatronischen Komponenten betrachtet. Daraus entstehen verschiedene Vor- aber auch Nachteile.

Vorteile:

- Auch unerfahrene Nutzer können mit den Systemen sehr gut arbeiten. Bereits nach kurzer Zeit ist ein funktionsfähiges mechatronisches System aufgebaut und programmiert, ohne dass Expertenwissen auf dem Gebiet der Elektrotechnik, der Mikrocontrollertechnik oder der Softwareentwicklung benötigt wird.
- Die vorgegebene mechatronische Hardware reduziert die Gefahr des fehlerhaften Anschließens von Sensoren und Aktoren, so dass eine Zerstörung von Bauteilen vermieden wird.

- Innerhalb der Programmierumgebungen wird mit den physikalischen Werten der Sensoren gearbeitet, so dass der Vorgang der Signalwandlung durch den Mikrocontroller und die Eigenschaften analoger und digitaler Signale vom Nutzer nicht verstanden werden müssen.
- Quellcodeerzeugung, Übersetzung und Herunterladen auf den Mikrocontroller erfolgen über einen einfachen Tastendruck. Die hierfür benötigten Werkzeuge sind vorinstalliert und müssen vom Nutzer nicht parametriert werden.

Nachteile:

- Die vorgestellten Systeme sind plattformabhängig. Für den Zentralbaustein, welcher den Mikrocontroller beinhaltet, ist keine Auswahl möglich.
- Der Zentralbaustein kann nur bestimmte Signalarten verarbeiten, wodurch nur speziell für ihn entwickelte Sensoren und Aktoren genutzt werden können.
- Für die Anbindung nicht unterstützter Sensoren und Aktoren ist Expertenwissen auf dem Gebiet der Elektronik, der Signalverarbeitung und der Softwareentwicklung notwendig.
- In den Programmierblöcken des Ablaufdiagramms sind lediglich geringfügige Anpassungen ausgewählter Parameter möglich. Dies schränkt die Flexibilität in der Programmierung deutlich ein.
- Mit den Systemen kann auf motivierende Art Wissen auf dem Gebiet der Robotik und der Mechanik vermittelt werden, jedoch nicht auf Gebieten wie der Elektrotechnik oder der mikrocontrollernahen Programmierung.

2.4.4 SPS-Programmierung nach IEC 61131-3

Speicherprogrammierbare Steuerungen (SPS) müssen flexibel einsatzfähig sein, weshalb sie in die Betrachtung zum Stand der Technik einfließen. Allerdings ist ihr Hauptanwendungsgebiet der industrielle Bereich [NGL2000],

weshalb nur am Rande auf diese Systeme eingegangen wird. Eine SPS besteht meist aus einem zentralen Baustein, welcher die eigentliche Rechentechnik beinhaltet, und aus zusätzlichen Modulen, welche für die Herstellung von Kommunikationsverbindungen oder die Anbindung von Sensoren genutzt werden können. Die Schnittstellen der Module sind standardisiert, so dass industrielle Sensoren und Aktoren mit Standardschnittstellen problemlos angeschlossen werden können [DIN/IEC60381-1-1985; DIN/IEC60381-2-1980].

In der IEC 61131-3 [DIN61131-3-2003] wurden folgende Sprachen für die Programmierung von Speicherprogrammierbaren Steuerungen standardisiert:

- Anweisungsliste (AWL) in textueller Form,
- Kontaktplan (KOP) in graphischer Form,
- Funktionsbausteinsprache (FBS) in graphischer Form,
- Ablaufsprache (AS) in graphischer und textueller Form und
- Strukturierter Text (ST) in textueller Form

Die graphischen Sprachen der Norm wurden in Abschnitt 2.2.3 vorgestellt.

Bekannte Vertreter von Entwicklungsumgebungen, welche auf den standardisierten Sprachen der IEC 61131-3 basieren, sind:

- STEP 7 [Sie2012] für Speicherprogrammierbare Steuerungen von Siemens
- CoDeSys [SSS2012b] unterstützt eine Reihe Speicherprogrammierbarer Steuerungen unterschiedlicher Hersteller [SSS2012a]

Um ein mechatronisches System zu entwickeln, muss zunächst die Speicherprogrammierbare Steuerung ausgewählt, zusammengestellt und an den PC angeschlossen werden. Anschließend kann die Programmierung innerhalb der Entwicklungsoberfläche beginnen. Dabei kommen die unterschiedlichen Sprachen zum Einsatz, wobei sich für zeitliche und bedingungsgebundene Abläufe eher die Ablaufsprache eignet, wohingegen die anderen

Sprachen eher für Berechnungen und logische Operationen eingesetzt werden.

Ist die Software für das System entwickelt, kann diese in wenigen einfachen Schritten übertragen werden.

Kritische Betrachtung

Speicherprogrammierbare Steuerungen kommen im Hobbybereich eher selten zum Einsatz, jedoch lassen sich aus dem Aufbau ihrer Bestandteile verschiedene Vor- und Nachteile ableiten.

Vorteile:

- Sensoren und Aktoren mit standardisierten Schnittstellen können einfach angebunden werden.
- Die Programmierung ist sehr flexibel, da unterschiedliche Sprachen zur Verfügung stehen.
- In der Software können auch komplexe Algorithmen umgesetzt werden, so dass eine hohe Flexibilität in der Programmierung gewährleistet ist.
- Um das erzeugte Mikrocontrollerprogramm zu übersetzen und auf das System zu laden, sind nur wenige einfache Schritte durch den Nutzer auszuführen. Dabei muss keine Anpassungen der genutzten Werkzeuge vorgenommen werden.

Nachteile:

- Nichtstandardisierte Sensoren und Aktoren benötigen zusätzliche Hardware oder können unter Umständen gar nicht genutzt werden. Der Nutzer benötigt daher Expertenwissen auf dem Gebiet der Elektrotechnik.
- Die Symbolik und die Bezeichnungen der normierten Programmiersprachen sind für Nutzer mit Expertenwissen gedacht. Unerfahrene Nutzer können nicht ohne weiteres mit diesen arbeiten.

- Der Nutzer muss die Theorie zur Signalwandlung am Mikrocontroller sowie die hierzu gehörigen Eigenschaften von Datentypen verstanden haben.
- Soll die gesamte Flexibilität in der Programmierung genutzt werden, müssen dem Nutzer alle Programmiersprachen vertraut sein. Dies ist vor allem bei den textuellen Sprachen eine Einstiegsbarriere.

2.4.5 Modulare Elektronikbausteine mit graphischer Ablaufprogrammierung

Die in diesem Abschnitt vorgestellten Systeme sind vergleichbar mit den unter Abschnitt 2.4.4 beschriebenen Speicherprogrammierbaren Steuerungen. Auch hier stehen zentrale Bausteine zur Verfügung, welche einen Mikrocontroller und die zugehörigen Programmier- und Treiberschaltkreise beinhalten. Auch können Sensoren und Aktoren über Erweiterungsmodule an das Zentralmodul angeschlossen werden. Allerdings sind die Erweiterungsmodule stärker auf den Einsatz im eingebetteten Bereich ausgelegt, so dass hier gewöhnlich eine breite Auswahl unterschiedlicher Module zur Verfügung steht, welche beispielsweise unterschiedliche Ein- und Ausgabegeräte beinhalten. Von den in Abschnitt 2.4.3 vorgestellten Systemen unterscheiden sich die Systeme dieses Abschnitts darin, dass hier noch keine Sensoren und Aktoren in die Elektronikbausteine integriert sind.

Zwei weit verbreitete Vertreter sind

- myAVR [Las2012] und
- FlowCode [Mat2011b].

Die Programmierung erfolgt bei diesen Systemen graphisch in zwei Ebenen. In der oberen Ebene wird ein Programmablaufplan nach DIN 66001 [DIN66001-1983] oder ein UML-Klassendiagramm nach ISO 19505 [ISO19505-1-2012] erstellt. Viele Objekte besitzen bereits ihre vollständige Funktionalität, so dass nur noch einige Parameter angepasst werden müssen. Anderen Objekten muss jedoch in der zweiten Ebene Quellcode zugewiesen werden, um sie mit Funktionalität zu versehen.

Kritische Betrachtung

Die vorgestellten Systeme sind vorrangig für professionelle Nutzer gedacht [Mat2012b]. Im Vergleich zur rein textuellen Programmierung erfolgt hier lediglich eine bessere und übersichtlichere Darstellung, wodurch der fortgeschrittene Nutzer deutliche Zeitersparungen in der Entwicklung hat. Für unerfahrene Nutzer ist das System ungeeignet, da

- textuelle Programmierfähigkeiten benötigt werden,
- Signalwandlungsprozesse vom Nutzer verstanden sein müssen,
- die graphischen Programmiersprachen nach DIN beherrscht werden müssen und
- der Nutzer Expertenwissen auf dem Gebiet der Elektrotechnik besitzen muss, um nicht unterstützte Sensoren und Aktoren anbinden zu können.

2.4.6 Zusammenfassung

Im Rahmen dieser Arbeit besteht das Ziel, die Entwicklung mikrocontrollerbasierter mechatronischer Systeme möglichst flexibel und plattformunabhängig zu gestalten. Trotz dieser beiden Vorgaben sollen die Konzepte eine einfache Anwendbarkeit für Nutzer ohne domänenspezifisches Expertenwissen ermöglichen.

In diesem Abschnitt wurden verschiedene Produkte vorgestellt, welche die Entwicklung mechatronischer Systeme vereinfachen sollen. Es erfolgte eine Einteilung in Kategorien, deren Beitrag zur einfachen Nutzbarkeit, Flexibilität und Plattformunabhängigkeit untersucht wurde.

Zusammengefasst sind im Bereich der Vereinfachung der Entwicklung die folgenden Ansätze erkennbar:

- Vereinfachter Aufbau der Mechanik/Elektronik durch modulare Hardware
- Vereinfachte Softwareentwicklung durch Nutzung graphischer Entwicklungsumgebungen und graphischer Programmiersprachen

- Vereinfachte Programmierung durch Bereitstellung entsprechender Infrastruktur (Toolchain)

Flexibilität wird durch folgende Maßnahmen ermöglicht:

- Flexibler Anschluss von Sensoren/Aktoren durch standardisierte Schnittstellen
- Nutzung textueller Sprachen

Wie bereits in Abschnitt 2.2.1 dargestellt, ist bei mikrocontrollerbasierten mechatronischen Systemen nicht mit einer vollständigen Plattformunabhängigkeit zu rechnen. Lediglich die Portierbarkeit bestimmter Programmbestandteile auf andere Plattformen kann mehr oder weniger flexibel gestaltet sein. Folgende Ansätze wurden identifiziert:

- Trennung von Plattformsoftware und Applikationssoftware, so dass der Teil der Applikationssoftware plattformübergreifend genutzt werden kann
- Unterstützung mehrerer Werkzeugketten innerhalb der Entwicklungsumgebung

Im folgenden Abschnitt werden diese Punkte noch einmal kritisch betrachtet.

2.5 Defizite bestehender Systeme

Zielstellung dieser Arbeit ist die Untersuchung und Entwicklung von Konzepten, welche die folgenden Voraussetzungen erfüllen:

- Nutzer ohne domänenspezifisches Expertenwissen (Elektronik- und Softwareentwicklung) sollen mit Hilfe der Konzepte mikrocontrollerbasierte mechatronische Systeme entwickeln können.
- Die Konzepte sollen eine möglichst große Flexibilität bei der Hardware- und Softwareentwicklung ermöglichen.

- Die Konzepte der Softwaremodellierung sollen eine möglichst plattformunabhängige Entwicklung ermöglichen.

Um die in Abschnitt 2.4 beschriebenen Systeme diesbezüglich näher vergleichend zu betrachten, werden diese im Folgenden anhand der genannten Kriterien eingeordnet.

2.5.1 Flexibilität der Hardware vs. Expertenwissen

Zunächst wird die Hardwareseite betrachtet. In Abbildung 18 ist daher die Flexibilität der Hardwareentwicklung in Abhängigkeit zum benötigten domänenspezifischen Expertenwissen dargestellt.

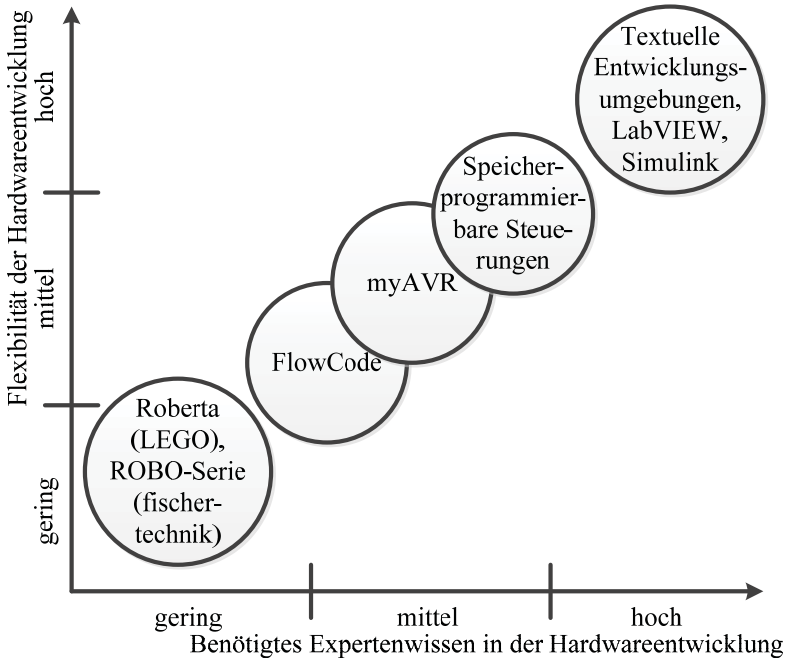

Abbildung 18: Darstellung der Flexibilität während der Hardwareentwicklung in Abhängigkeit des benötigten domänenspezifischen Expertenwissens

Dabei beinhaltet die Betrachtung der Flexibilität der Hardwareentwicklung zum einen die hardwareseitige Plattformunabhängigkeit des genutzten informationstechnischen Systems und zum anderen die freie Wählbarkeit der einsetzbaren Sensoren und Aktoren. Je besser diese beiden Punkte erfüllt sind, desto flexibler kann die Hardware an den vorgegebenen Prozess oder an zusätzlich vorgegebene Randbedingungen angepasst werden.

Abbildung 18 zeigt, dass, bei Nutzung der vorgestellten Systeme, eine große Flexibilität während der Hardwareentwicklung mit einem hohen Maß an benötigtem Expertenwissen einhergeht. Um die Gründe hierfür zu erkennen, wurde der Aufbau der Hardware der in den Abschnitten 2.4.1 bis 2.4.5 vorgestellten Systeme untersucht.

Es konnten drei Kategorien identifiziert werden:

- Gebrauchsfertige Module mit umfassender Funktionalität werden bereitgestellt. Hier ist kein Expertenwissen für den Aufbau des mechatronischen Systems notwendig. Allerdings können nur Hardwarefunktionen realisiert werden, welche von den Entwicklern vorgesehen wurden. Die informationstechnische Hardwareplattform ist im Normalfall nicht austauschbar.
- Diese Produkte basieren auf der Bereitstellung standardisierter Schnittstellen. Meist stehen dabei auch mehrere Plattformen zur Auswahl zur Verfügung. Durch die Standardisierung der Schnittstellen wird die Entwicklung der mechatronischen Hardware vereinfacht, da ein breites Spektrum passender industrieller Sensoren und Aktoren zur Verfügung steht. Allerdings werden vor allem im Hobbybereich oft preisgünstige Sensoren und Aktoren mit nichtstandardisierten Schnittstellen eingesetzt, welche in Verbindung mit Produkten der zweiten Ebene nicht ohne weitreichendes Expertenwissen nutzbar sind.

- Produkte der dritten Ebene stellen faktisch keine Werkzeuge zur Unterstützung der Hardwareentwicklung bereit. Um ein mechatronisches System zu entwickeln, wird ein hohes Maß an Expertenwissen vorausgesetzt. Mit diesem Wissen kann jedoch auch eine Vielzahl unterschiedlicher Funktionalitäten realisiert werden, da es kaum Einschränkungen bei der Plattform-, Sensor- und Aktorauswahl gibt.

In jeder der Realisierungsformen der Kategorien sind Flexibilität und benötigtes Expertenwissen eng miteinander verknüpft. Somit bedeutet eine Flexibilisierung auch immer eine Erhöhung des benötigten Expertenwissens.

2.5.2 Flexibilität der Software vs. Expertenwissen

In der Untersuchung der existierenden Entwicklungssysteme zeigte sich, dass auch im Bereich der Softwareentwicklung ein vergleichbarer Zusammenhang zwischen benötigtem domänenspezifischen Expertenwissen und Flexibilität existiert. In Abbildung 19 ist dieser Zusammenhang graphisch dargestellt.

Wie auch in Abschnitt 2.5.1 lassen sich drei Gruppen identifizieren:

- Es wird eine einfach verständliche, graphische Programmiersprache auf einer hohen abstrakten Ebene genutzt, wodurch der Nutzer nahezu kein Expertenwissen benötigt. Für die Programmierung werden modulare Programmierblöcke genutzt, welche durch den Nutzer lediglich parametriert und in der richtigen Reihenfolge angeordnet werden müssen. Die Funktionalität der entstehenden Software beschränkt sich hierbei auf die von den Entwicklern vorgesehenen Programmierblöcke.
- Zur Programmierung werden mehrere standardisierte, graphische Sprachen kombiniert. Dadurch entsteht eine verbesserte Flexibilität, da sich der Nutzer entscheiden kann, in welcher Sprache er die gewünschte Funktionalität implementieren kann. Allerdings sind die meisten standardisierten Sprachen nicht intuitiv nutzbar, da sie sich aus der Informatik, der Elektrotechnik oder vergleichbaren Domänen ableiten. Grundlegendes Expertenwissen aus diesen Fachbereichen wird daher vorausgesetzt.

- Textuelle Programmiersprachen bieten die größtmögliche Flexibilität während der Softwareentwicklung. Allerdings wird hierfür ein umfassendes Expertenwissen in den unterschiedlichen angrenzenden Domänen benötigt.

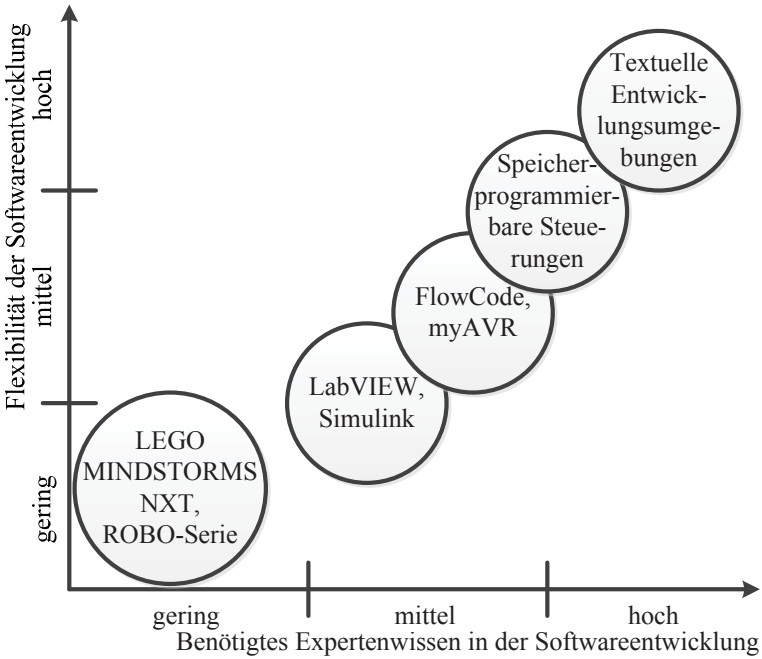

Abbildung 19: Darstellung der Flexibilität während der Softwareentwicklung in Abhängigkeit des benötigten domänenspezifischen Expertenwissens

2.5.3 Identifizierte Schwachstellen

Aus den Betrachtungen der Abschnitte 2.5.1 und 2.5.2 lassen sich einige Schwachstellen der existierenden Systeme identifizieren. In den Abbildungen dieser Abschnitte sind deutliche Korrelationen zwischen der Flexibilität und dem benötigten Expertenwissen erkennbar. Dies hat verschiedene Gründe:

- Bei den hardwareseitig unterstützten Systemen erfolgt die Unterstützung in Form vorgefertigter Module. Damit verbessert sich die Nutzbarkeit für Nichtexperten. Es verringert aber die Flexibilität und die Plattformunabhängigkeit.
- Unterstützung bei der Softwareentwicklung erhält der Nutzer über zwei Ansätze:
  - Wie bei der Hardware wird auch die Funktion der Software in Modulen verpackt, was wiederum die Flexibilität reduziert.
  - Andere Produkte ermöglichen eine Kombination unterschiedlicher Programmiersprachen, welche jedoch nur durch Nutzer mit Expertenwissen nutzbar sind.

Ziel dieser Arbeit ist es, eine möglichst hohe Flexibilität in der Hardware- und Softwareentwicklung zu ermöglichen und dennoch Nutzern mit nur geringem domänenspezifischem Expertenwissen die Entwicklung mechatronischer Systeme im Hobbybereich zu ermöglichen. In Abbildung 20 und in Abbildung 21 wird diese Zielstellung graphisch in die bereits existierenden Systeme eingeordnet.

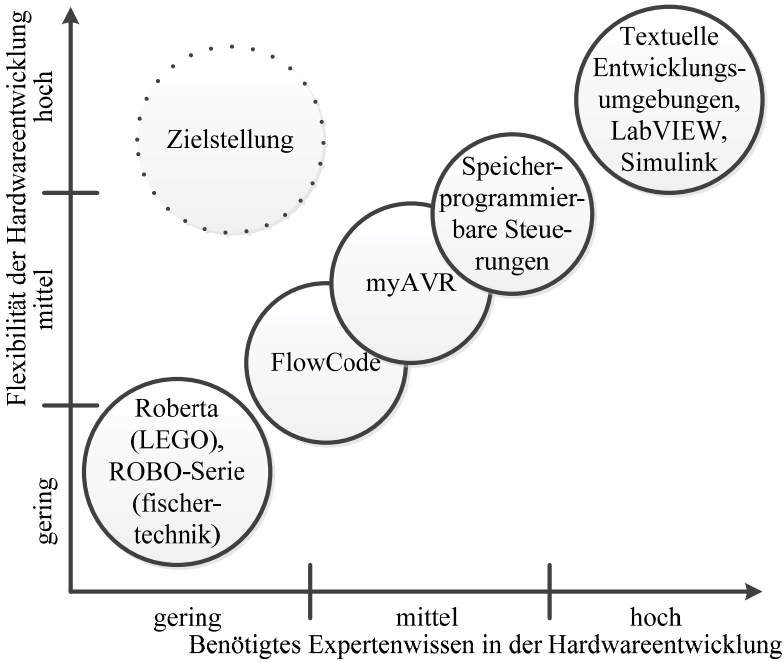

Abbildung 20: Einordnung der Zielstellung dieser Arbeit innerhalb der Darstellung der Flexibilität der Hardwareentwicklung in Abhängigkeit des benötigten Expertenwissens

Es ist zu erkennen, dass nicht vorgesehen ist, das obere Ende der Skala der Flexibilität beziehungsweise das untere Ende der Skala des benötigten Expertenwissens zu erreichen. Dies wär zwar anstrebenswert, ist aber nicht realistisch, da die Ziele offensichtlich untereinander in Konkurrenz stehen. In den folgenden Kapiteln werden Ansätze und Konzepte vorgestellt, mit welchen diese Zielstellung erreicht werden kann.

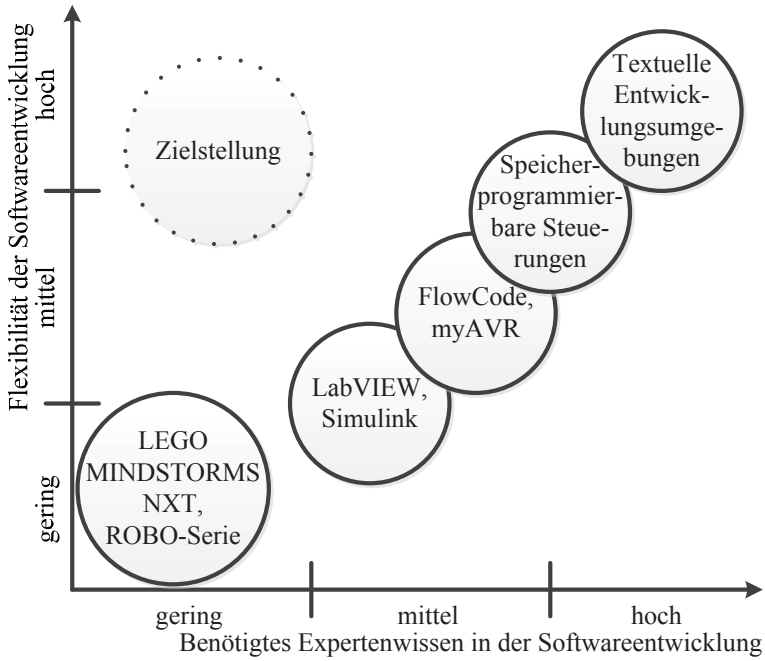

Abbildung 21: Einordnung der Zielstellung dieser Arbeit innerhalb der Darstellung der Flexibilität der Softwareentwicklung in Abhängigkeit des benötigten Expertenwissens

2.6 Zusammenfassung

In diesem Kapitel wurden einige Einblicke in die Grundlagen dieser Arbeit sowie in den Stand der Technik auf dem bearbeiteten Gebiet gegeben. Hierfür erfolgte zunächst die Einführung des Begriffs der mikrocontrollerbasierten mechatronischen Systeme mit einer Beschreibung ihres grundsätzlichen Aufbaus und des Entwurfsprozesses.

Der sich anschließende Abschnitt beinhaltete einige grundlegende Informationen zur Programmierung von Mikrocontrollern, wobei vor allem Hintergrundwissen über die modellbasierte und graphische Softwareentwicklung im Zentrum der Betrachtungen stand. In diesem Kontext wurden einige übliche graphische Modellierungssprachen beschrieben.

Da der Entwicklungsprozess der mechatronischen Systeme möglichst nutzerfreundlich gestaltet sein soll, wurden im sich anschließenden Abschnitt einige Grundlagen zur Usability-Evaluierung von Mensch-Maschine-Schnittstellen mit einem Kurzüberblick über die wichtigsten Evaluierungsmethoden vorgestellt.

Eine wichtige Recherche zum Stand der Technik erfolgte im Rahmen einer Untersuchung der wichtigsten und bekanntesten existierenden Entwicklungssysteme für mikrocontrollerbasierte mechatronische Systeme. Als wichtigstes Defizit konnte aus dieser Untersuchung abgeleitet werden, dass bei den bisher genutzten modulbasierten Ansätzen ein direkter Zusammenhang zwischen der Flexibilität der Entwicklung der Hard- und Software und dem hierfür benötigten Expertenwissen besteht. Um diese Verkopplung aufzubrechen, werden neue Konzepte und Ansätze zur Entwicklung mikrocontrollerbasierter mechatronischer Systeme untersucht.

3 EasyKit und EasyKit Starter

In den Jahren 2007 bis 2009 wurde gemeinsam mit der Technischen Universität München, der Festo Didactic GmbH & Co. KG, dem Verband Deutscher Maschinen- und Anlagenbau e.V., der digiraster Tetzner GmbH, der efm-systems GmbH, der KSB AG und der Pumpenfabrik Ernst Scherzinger GmbH & Co. KG das vom BMBF geförderte und vom PTKA (Projektträger Karlsruhe) betreute Kooperationsprojekt EasyKit durchgeführt. Die Projektpartner sind in Abbildung 22 noch einmal dargestellt. [Eas2011]

Abbildung 22: Konsortium des Projekts EasyKit

Im Verlauf des Projekts EasyKit wurde die gleichnamige Methodik zur Entwicklung modularer mechatronischer Systeme für industrielle Anwender geschaffen. Dabei werden verschiedene modulare Elektronikbausteine miteinander verbunden und über die Entwicklungsumgebung EasyLab mit Hilfe graphischer Programmiersprachen programmiert. Die Bausteine liegen in miniaturisierter Form vor, so dass sie trotz des modularen Aufbaus in nahezu jedes System integriert werden können. Die Entwicklungsumgebung EasyLab ist in erster Linie darauf ausgerichtet, eine möglichst plattformunabhängige, schnelle, einfache und flexible Programmierung zu ermöglichen, wodurch Entwicklungszeit gespart werden kann.

Die Einsetzbarkeit durch Nutzer ohne Expertenwissen stand in diesem Projekt noch nicht im Mittelpunkt. Allerdings stellte sich nach ersten Tests heraus, dass mit nur wenigen Veränderungen ein System geschaffen werden könnte, welches sogar im schulischen Umfeld einsetzbar wäre. Hardware und Software wurden für diesen Zweck weitestgehend vereinfacht und sind nun unter dem Namen „EasyKit Starter“ verfügbar [Fes2010].

Da beide Systeme aufeinander aufbauen und daher nur schwer voneinander getrennt betrachtet werden können, wird in diesem Kapitel zunächst die originale, industrielle Version von EasyKit und daran anschließend der EasyKit Starter vorgestellt.

Anmerkung des Autors

Wie bereits erwähnt, war EasyKit ein Kooperationsprojekt, weshalb die vorgestellten Ergebnisse aus dem Zusammenwirken mehrerer Personen entstanden. Um ein umfassendes Bild abzugeben, ist es jedoch notwendig, das gesamte Projekt darzustellen.

Das hauptsächliche Mitwirken des Autors war angesiedelt im Bereich

- der Entwicklung und Analyse möglicher Anwendungsfälle,
- der Hard- und Softwareentwicklung,
- der Analyse des beim Einsatz des Systems benötigten Expertenwissens und
- der Evaluierung im Rahmen von Nutzertests.

3.1 Entwicklung mit EasyKit

Wie bereits in Abschnitt 2.1.2 beschrieben und in Abbildung 5 dargestellt wurde, erfolgt die Entwicklung mechatronischer Systeme traditionell schrittweise. [BPK2009].

Auch wenn sich ein solcher sequentieller Arbeitsablauf positiv auf die Zuverlässigkeit des zu entwickelnden Systems auswirkt [Ste2005], erhöht sich hierdurch die Entwicklungszeit. Dies gilt vor allem, wenn die mechatronischen Systeme lediglich in geringen Stückzahlen produziert werden sollen.

Ein weiteres Problem ist, dass mindestens drei Experten an der Entwicklung beteiligt sein müssen, da für jeden Schritt Expertenwissen auf dem jeweiligen Gebiet benötigt wird, um die Zuverlässigkeit des zu entwickelnden mechatronischen Systems zu gewährleisten. Dieser Ressourcenbedarf erhöht die Kosten für das entstehende Produkt und ist oft für kleinere Unternehmen gar nicht ohne kostenintensive, externe Unterstützung aufzubringen.

Um hierfür einen Lösungsansatz zur Verfügung zu stellen, wurde das EasyKit-Konzept entwickelt. Es basiert vor allem auf der Vereinfachung des Schaltungsentwurfs und der Programmierung. Die Vereinfachung des Schaltungsentwurfs wird durch eine Modularisierung erreicht. Dabei werden häufig genutzte Schaltungen in Form von Elektronikmodulen bereitgestellt. Die Programmierung erfolgt durch graphische Modellierung in der Entwicklungsumgebung EasyLab. Aus dem so erstellten Modell wird von EasyLab der Quellcode des eigentlichen Mikrocontrollerprogramms generiert.

Durch diese Vereinfachungen ist es möglich, den Anteil an benötigtem Expertenwissen zu reduzieren, da dies bereits in die entwickelten Module eingeflossen ist. Zudem kann der Entwicklungsprozess im Rahmen eines koordinierten Vorgehens so parallelisiert werden, wie in Abbildung 23 dargestellt.

Durch den modularen Aufbau können die Entwicklungszeiten auf zwei Ebenen reduziert werden. Zum einen wird die eigentliche Schaltungs- und Codeentwicklung beschleunigt, zum anderen wird aber auch die Gesamtentwicklungszeit durch die Möglichkeit der Parallelisierung der Arbeitsschritte verringert.

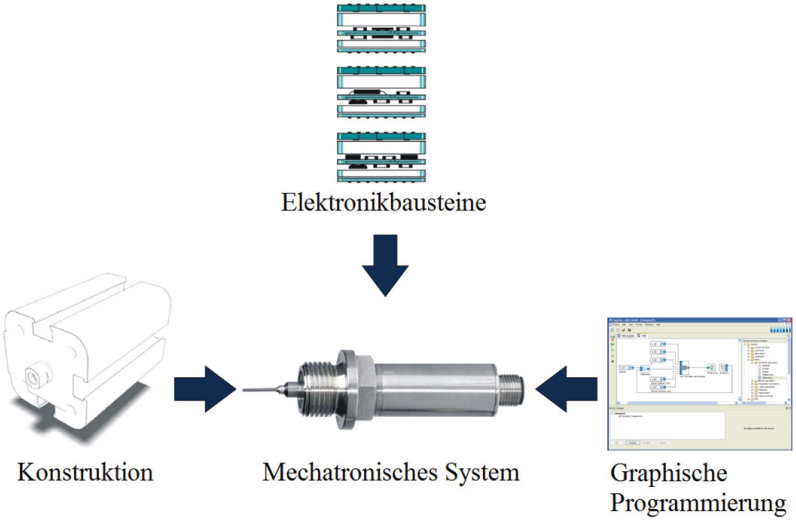

Abbildung 23: Paralleler Ablauf der Entwicklung eingebetteter Systeme mit Hilfe modularisierter Elektronik und graphischer Programmierung

3.1.1 EasyKit Elektronikmodule

Wie bereits beschrieben, werden die benötigten elektronischen Schaltungen mit Hilfe einzelner Module erstellt, wovon einige in Abbildung 24 dargestellt sind. Diese werden je nach zu erfüllender Aufgabe zu einem Stapel zusammengesteckt, welcher möglichst geringe Abmessungen besitzen soll, da er in mechanische Systeme eingebettet wird. Im Normalfall besteht ein solcher Stapel aus einem Spannungsversorgungsmodul, einem CPU-Modul und mindestens einem Schnittstellenmodul.

Wie der Name schon vermuten lässt, wird im Spannungsversorgungsmodul eine von außen angelegte Gleichspannung in eine vom Mikrocontroller nutzbare Versorgungsspannung gewandelt. Des Weiteren befinden sich an diesem Modul Steckverbindungen, über welche der Stapel mit einem PC oder anderen Systemen verbunden werden kann, um so den Mikrocontroller programmieren oder ihn in ein Bussystem einbinden zu können.

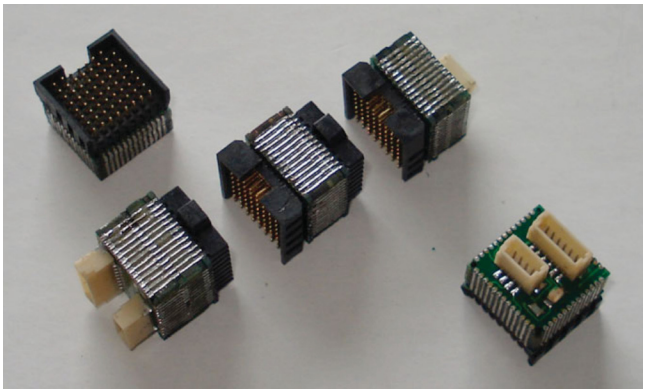

Abbildung 24: EasyKit Elektronikmodule

Das CPU-Modul beinhaltet den Mikrocontroller mit dem zugehörigen Quarzoszillator sowie zusätzliche Treiberschaltkreise zur Kommunikation. Es wurden bereits verschiedene Prozessoren der Firmen Microchip und ATMEL in CPU-Module integriert. Für die Kommunikation kann zwischen Modulen mit integrierten Treibern für RS232, RS485 und CAN gewählt werden.

Die Schnittstellenmodule können zum Teil miteinander kombiniert werden. Sie stellen Steckverbindungen bereit, an welche Sensoren und Aktoren mit unterschiedlichen Schnittstellen angeschlossen werden können. Dabei werden verschiedene industrielle Standardschnittstellen, wie zum Beispiel Spannungssignale von 0…10 V, abgedeckt. Weitere weit verbreitete Schaltungen, wie zum Beispiel eine Auswerteelektronik mit Wheatstonebrücke und hochauflösendem Analog-Digital-Wandler für PT100-Messspitzen oder Treiber für induktive Lasten wie Relais, sind ebenso verfügbar.

Da bei solchen modularen Systemen nicht sämtliche möglichen Schnittstellen berücksichtigt werden können, besteht die Möglichkeit, einzelne Modulkomponenten, wie Rahmen und Trägerplatinen, mit selbstentwickelten Schaltungen zu bestücken. Die für diesen Schritt benötigten Hintergrundinformationen über die mechanischen und elektrischen Schnittstellen sind im VDMA Einheitsblatt 66305 beschrieben [VDMA66305-2003]. Des Weiteren sind Layoutvorlagen für die Trägerplatinen verfügbar, welche beispiels-

weise in der PCB-Software „Eagle“ genutzt werden können. So ist es nicht nur möglich, weitere Schnittstellen zu realisieren, sondern der Nutzer kann auch neue Mikrocontroller integrieren.

Ein solches modulbasiertes Hardware-System bietet verschiedene Vorteile:

- Durch die Nutzung standardisierter Module erhöht sich die Zuverlässigkeit des Systems.
- Die Entwicklungszeit wird erheblich reduziert, was vor allem bei Systemen mit geringen Stückzahlen oder in der Prototypenentwicklung eine erhebliche Kostenreduzierung ermöglichen kann.
- Das für einen Entwicklungsprozess benötigte Expertenwissen auf dem Gebiet der Elektronikentwicklung wird erheblich reduziert, da nur noch ein grundlegendes Verständnis für die Funktionen des Moduls, jedoch nicht für seine einzelnen elektronischen Bestandteile vorausgesetzt wird.
- Da die Module neben standardisierten Schnittstellen auch die Anbindung anderer oft genutzter Hardware ermöglichen, ist mit der elektronischen Hardware von EasyKit bereits eine relativ hohe Flexibilität erkennbar.

Jedoch resultieren aus dem modularen Aufbau auch einige Nachteile:

- Es kann nicht für jede existierende Schnittstelle ein Schnittstellenmodul zur Verfügung gestellt werden. Der Einsatz von Sensoren mit Standardschnittstellen ist daher anzustreben. Da Sensoren aus dem Low-Cost-Segment solche Schnittstellen oft nicht besitzen, können sich die Kosten des Endproduktes erhöhen.
- Können keine unterstützten Schnittstellen genutzt werden, muss ein neues Modul angefertigt werden, welches die entsprechenden Spezifikationen besitzt. In diesem Fall ist wiederum Expertenwissen auf dem Gebiet der Elektronikentwicklung erforderlich.

Die Betrachtung der Hardware von EasyKit zeigt, dass der Einsatz des Systems wohl durchdacht sein muss. Ein positiver Nutzen kann vor allem bei

der Entwicklung von Prototypen und Kleinserien erzielt werden. Dies gilt insbesondere wenn weitere Randbedingungen eingehalten werden müssen, wie zum Beispiel eine möglichst gute Plattformportierbarkeit oder Größenbeschränkungen.

3.1.2 EasyLab Entwicklungsumgebung

Für die Programmierung des im CPU-Modul enthaltenen Mikrocontrollers wurde im Rahmen des BMBF-Projekts die graphische Programmierumgebung EasyLab entwickelt. Dabei werden zwei hierarchisch angeordnete, graphische Programmiersprachen genutzt, welche auch in der SPS-Programmierung Anwendung finden und im Rahmen der Norm IEC 61131 standardisiert wurden [DIN61131-3-2003]. Die obere Ebene der Programmierung erfolgt durch die Erstellung eines Diagramms in der Ablaufsprache (Abbildung 25), während die untere Ebene durch einen Datenflussplan in Funktionsbausteinsprache (Abbildung 26) realisiert ist.

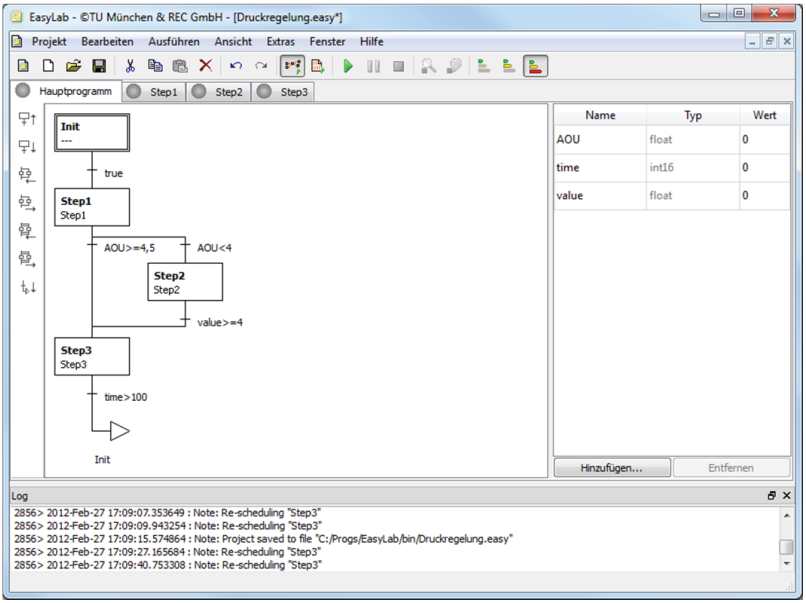

Abbildung 25: Ablaufdiagramm in EasyLab

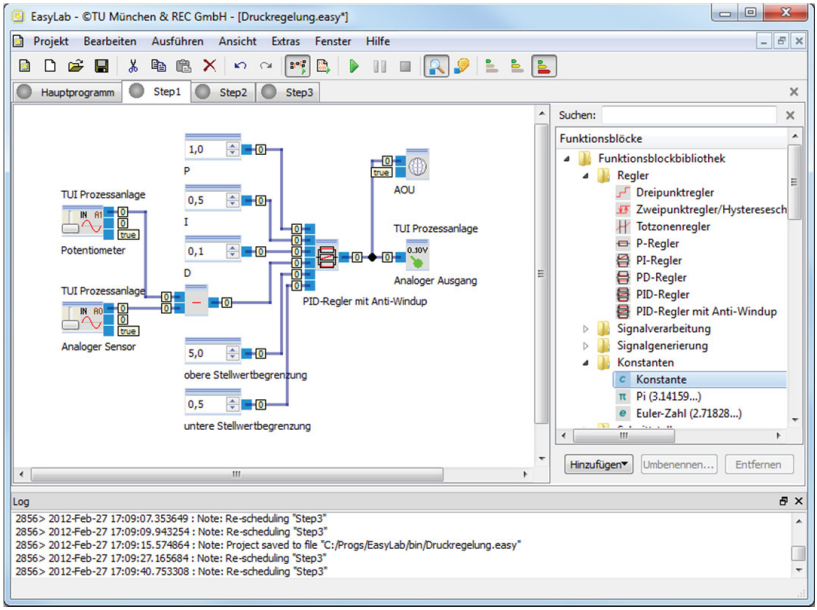

Abbildung 26: Funktionsbausteinsprache in EasyLab

Diese beiden Sprachen wurden gewählt, da sie zum einen vergleichsweise leicht verständlich sind und zum anderen durch ihre Kombination eine Realisierung kombinierter ereignisgesteuerter Abläufe und datenflussorientierter Berechnungen ermöglicht wird. Diese Kombination ermöglicht eine hohe Flexibilität in der Programmierung. [BGH2010]

Wie in den beiden Abbildungen dargestellt, ist die Benutzeroberfläche grundsätzlich dreigeteilt. Im oberen linken Teil findet die eigentliche Programmierung mit Hilfe der beiden graphischen Programmiersprachen statt. Im unteren Teil befindet sich das Protokoll, welches Informationen über den aktuellen Systemzustand beinhaltet. Rechts befinden sich Variablen oder Bibliotheken, welche für die Modellierung genutzt werden.

Das Diagramm der Ablaufsprache besteht aus Zuständen, welche mit gerichteten Verbindern gekoppelt sind. Für die Verbinder werden vom Nutzer logische Ausdrücke vorgegeben, welche festlegen, unter welchen Bedingun-

gen der Wechsel von einem Zustand in einen folgenden erfolgt. Für die Erstellung dieser Bedingungen stehen die Variablen aus dem Fenster auf der rechten Seite der Oberfläche zur Verfügung. Den Variablen können innerhalb der Programmierung im Datenflussplan Werte zugewiesen werden. Ist keine Bedingung am Ende eines Zustands erfüllt, so wiederholt sich der aktuelle Zustand. Der Nutzer kann außerdem eine Mindestausführungsdauer für jeden einzelnen Zustand einstellen.

Benötigt das Ausführen des entsprechenden Zustands weniger als die voreingestellte Zeit, so wartet das Programm vor der nächsten Ausführung so lange, bis die Zeit abgelaufen ist.

Jedem Zustand des Ablaufdiagramms muss eine Funktion zugewiesen werden. Hierfür muss diese Funktion mit Hilfe der Funktionsbausteinsprache im Datenflussplan modelliert werden. Ein so erzeugter Datenflussplan kann anschließend einem Zustand im Ablaufdiagramm zugeordnet werden. In Abbildung 25 und in Abbildung 26 wurde dies beispielhaft für die Datenflusspläne Step1 bis Step3 durchgeführt.

Funktionsbausteinbibliothek

Wie in Abbildung 26 zu erkennen ist, befindet sich die Funktionsbausteinbibliothek während der Programmierung im Datenflussplan auf der rechten Seite des Arbeitsbereichs. Von hier werden die einzelnen Funktionsbausteine per Drag-and-Drop in das links davon befindliche Hauptfenster gezogen und miteinander verbunden. Somit entsteht ein Datenflussplan, wie er beispielsweise aus MATLAB Simulink [Mat2011a] bekannt ist.

In EasyLab stehen für die Programmierung in der Funktionsbausteinsprache

- generische Funktionsbausteine,
- erweiterte Funktionsbausteine und
- plattformspezifische Funktionsbausteine zur Verfügung [BGH2010].

Mit Hilfe der generischen Funktionsbausteine können mathematische Berechnungen, logische Verknüpfungen, Vergleiche und viele weitere einfache Abläufe realisiert werden. Die Funktionen dieser Funktionsbausteine stehen auch in den meisten textuellen Programmiersprachen so oder in vergleichbarer Form zur Verfügung.

Die erweiterten Funktionsbausteine beinhalten verschiedene nicht generische Funktionen, wie zum Beispiel die Visualisierung von Daten während der Laufzeit innerhalb von EasyLab oder auch Funktionsbausteine zum Lesen und Schreiben global genutzter Variablen. Ein wichtiger Teil der erweiterten Funktionsbausteine ist zudem eine Bibliothek für die Nutzung im Bereich der Automatisierungs- und Regelungstechnik. Diese beinhaltet verschiedene Filter, Regler und Signalgeneratoren. [BA2010b]

Mit Hilfe der Filterfunktionsbausteine können Signale verarbeitet werden, indem beispielsweise Mittelwertbildungen, Glättungen oder auch mathematische Transformationen durchgeführt werden. Als Regler stehen Zweipunktregler mit optionaler Hysterese, Dreipunktregler, Totzonenregler und klassische PID-Regler mit optionalem Anti-Windup zur Verfügung. Die Signalgeneratorfunktionsbausteine können Rampen-, Sägezahn-, Sprung-, Sinus- und pulsweitenmodulierte Signale erzeugen. Die Funktionsbausteine der Automatisierungs- und Regelungstechnikbibliothek sind so implementiert, dass der Nutzer lediglich Grundlagenwissen und kein umfassendes Hintergrundwissen über die Umsetzung im Quellcode der einzelnen Funktionsbausteine benötigt. Der PID-Regler ist beispielsweise über die üblichen Proportional-, Integral- und Differentialkoeffizienten zu parametrieren. [BA2010b]

Während die bisher beschriebenen Funktionsbausteine reine Softwarefunktionalität zur Verfügung stellen, ermöglichen die plattformspezifischen Funktionsbausteine die Manipulation und das Auslesen der Hardware. Zu den wichtigsten Funktionen gehören dabei das Lesen und Schreiben der digitalen Ein- und Ausgänge sowie das Lesen der digitalen Daten der Analog-Digital-Umsetzer (ADU) des Mikrocontrollers. Je nach Funktionsumfang des genutzten Mikrocontrollers stehen weitere Funktionen zur Verfügung, wie zum Beispiel das Erzeugen eines pulsweitenmodulierten Signals an einem Ausgang, welches beispielsweise für die Ansteuerung von Motoren genutzt werden kann.

Die Namen der Funktionsbausteine und ihrer Signalein- und Signalausgänge wurden so gewählt, dass sie deren Funktionen möglichst genau beschreiben. Da jedoch zum einen das Programm von Nutzern mit unterschiedlichem Vorwissen genutzt werden soll und zum anderen bei komplexeren Funkti-

onsbausteinen der Name nicht umfassend beschreibungsfähig sein kann, steht dem Nutzer eine Hilfeseite zur Verfügung, auf welcher die Funktionen der Funktionsbausteine sowie ihrer Signalein- und Signalausgänge beschrieben sind.

Simulation und Debugging

Nachdem das Mikrocontrollerprogramm mit Hilfe der oben beschriebenen graphischen Programmiersprachen entwickelt wurde, kann seine Funktion mit der Simulationsfunktion von EasyLab getestet werden. EasyLab simuliert dabei die Berechnungen des Mikrocontrollers und stellt diese innerhalb der graphischen Programmieroberfläche dar. Der Nutzer kann das Signalverhalten der digitalen und analogen Eingänge des Mikrocontrollers direkt in der Programmieroberfläche beeinflussen. Somit ist eine Simulation des Verhaltens von Sensoren möglich, welche an den Eingängen angeschlossen sind.

Verhält sich das Programm wie gewünscht, kann es auf den Mikrocontroller geladen werden. Erfahrenen Nutzern ist es möglich im Auswahldialog für das Brennen auf den Mikrocontroller verschiedene zusätzliche Parameter anzupassen. Für die üblichen Anwendungen müssen diese jedoch nicht verändert werden. In diesem Dialog ist es zudem möglich, das Debugging zu aktivieren, wodurch EasyLab nach dem Übertragen der Daten auf den Mikrocontroller in den Debugmodus wechselt. Dabei werden im Datenflussplan an allen Funktionsbausteinen die Werte der Ein- und Ausgänge angezeigt. So kann getestet werden, wie sich der programmierte Mikrocontroller verhält, während er in das System eingebettet ist.

Plattformunabhängigkeit

Wurde in EasyLab Software für einen unterstützten Mikrocontroller entwickelt, so kann diese auch für andere unterstützte Controller genutzt werden. Lediglich die plattformspezifischen Funktionsbausteine müssen ersetzt werden.

Die Entwicklung der Software kann grundsätzlich auch für Mikrocontroller erfolgen, welche nicht von der Hardware von EasyKit unterstützt werden. Hierfür muss lediglich die entsprechende Werkzeugkette (Toolchain) in

EasyKit integriert und der Zugriff auf die hardwarespezifischen Funktionen implementiert werden.

3.1.3 Fazit

Mit EasyKit können industrielle Anwender bei der Entwicklung mechatronischer Systeme effektiv unterstützt werden. Durch die Nutzung modularer Elektronikhardware und graphischer Programmiersprachen mit modularen Softwarebausteinen ist nur wenig Expertenwissen aus dem Bereich der Elektronik- und Softwareentwicklung erforderlich und die Entwicklung wird erheblich beschleunigt, da Mechanik, Elektronik und Software parallel entwickelt werden können. Dabei ist vor allem die Flexibilität der Programmierung bemerkenswert, welche durch die Kombination der Ablaufsprache und der Funktionsbausteinsprache erreicht wird. Weitere Bestandteile der Software wie der Simulationsmodus und das Onlinedebugging erleichtern die Inbetriebnahme des zu entwickelnden mechatronischen Systems.

Allerdings zeigt sich an verschiedenen Stellen, dass das System auf industrielle Nutzer mit technischem Hintergrundwissen ausgelegt ist. Um das System effektiv nutzen zu können, wird zumindest ein grundlegendes ingenieurmäßiges Denken vorausgesetzt.

Wie bereits beschrieben, soll im Rahmen dieser Arbeit eine weitestgehend flexible und plattformunabhängige Programmierung für Nutzer ohne Expertenwissen ermöglicht werden. Die folgenden Konzeptansätze liefern hierzu einen wichtigen Beitrag:

- Erhöhte Plattformunabhängigkeit/Plattformportierbarkeit durch
  - Unterstützung unterschiedlicher Plattformen verschiedener Hersteller,
  - Möglichkeit der Integration neuer Plattformen in Hard- und Software und
  - Trennung von Plattformsoftware- und Applikationssoftwarebestandteilen.
- Erhöhte Flexibilität der Hardwareentwicklung durch
  - Unterstützung standardisierter Schnittstellen und

- zusätzliche Unterstützung verschiedener nichtstandardisierter Schnittstellen.

- Erhöhte Flexibilität der Softwareentwicklung durch
  - Kombination ereignisgesteuerter und datenflussorientierter Programmierung.
- Vereinfachte Programmierung durch
  - Auswahl gut verständlicher graphischer Programmiersprachen.

Weitere Informationen zur industriellen Version von EasyKit sind in verschiedenen Veröffentlichungen zu finden [BA2010b; BA2010c; BGH2010; Eas2011].

3.2 EasyKit Starter

Von verschiedenen Technikdidaktikern wird angeführt, dass technische Themengebiete aus dem Bereich der Mechatronik im Schulunterricht unterrepräsentiert sind [Mar2006] und erst zu spät unterrichtet werden, da die Technikbegeisterung lediglich bis etwa zur siebenten Klasse geweckt werden kann [Hei2011].

Durch die oben beschriebene einfache Nutzbarkeit des Systems konnten auch Schüler höherer Klassenstufen sowie Berufsschüler als mögliche Zielgruppen einbezogen werden. Allerdings erschien die Nutzung der elektronischen Module im Rahmen des Schulunterrichts nur wenig sinnvoll, da die Anforderungen hier anders gesetzt werden müssen. So ist beispielsweise eher auf eine flexible und sichere Handhabbarkeit zu achten, als auf die Miniaturisierung, wie sie bei der industriellen Version gefordert ist. Aufgrund dieser Anforderungen wurden ein EasyKit Starterboard und ein Applikationsboard entwickelt, welche später in diesem Abschnitt näher beschrieben werden. Außerdem wurden in EasyLab einige Veränderungen vorgenommen, um die Software für die Zielgruppe einfacher und verständlicher zu gestalten. Um den Schülern eine erste Einführung in das Fachgebiet der Mechatronik zu geben und somit den Einstieg in die Nutzung des Systems zu

erleichtern, steht ein „web-based Training“ bereit. In den folgenden Abschnitten werden diese Bestandteile des EasyKit Starters beschrieben und ihre Funktionen erklärt.

3.2.1 EasyKit Starterboard

Das EasyKit Starterboard beinhaltet neben den Funktionen der industriellen Version von EasyKit einige Erweiterungen, welche die ersten Schritte zur Nutzung des Systems erleichtern. Die wichtigste Veränderung zur industriellen Version von EasyKit ist sicher, dass mit dem Starterboard bereits einige kleine Applikationen möglich sind. Das Board besitzt drei unterschiedlich farbige LEDs (lichtemittierende Dioden) und drei Taster, mit welchen beispielsweise eine einfache Ampelschaltung realisiert werden kann. Optional kann eine Flüssigkristallanzeige (LCD – Liquid Crystal Display) in der rechten oberen Ecke bestückt werden. In Abbildung 27 ist das Starterboard dargestellt.

Sensoren und Aktoren werden nicht mehr wie bei der industriellen Version über Steckverbindungen an das System angeschlossen, sondern über die Schraubkontaktleiste im unteren Bereich des Starterboards. Die Schnittstellenmodule der industriellen Version kommen hier dennoch zum Einsatz. Sie werden oberhalb der Schraubkontaktleiste auf das Board gesteckt und weisen jedem Schraubkontakt eine bestimmte Funktion zu, indem sie zum Beispiel Spannungsteiler, Tiefpassfilter oder Treiber für induktive Lasten beinhalten. Des Weiteren sind in ihnen Schutzschaltungen integriert, welche beispielsweise gegen Überspannungen oder Verpolung schützen. Abgesehen von den Schnittstellenmodulen wurde beim Starterboard auf die modulare Bauweise verzichtet, was jedoch im schulischen Anwendungsfall keine Einschränkung für die Nutzer darstellt, da die Flexibilität in der Nutzung unterschiedlicher Mikrocontroller und Bussysteme hier nicht benötigt wird. Während in den Modulen der industriellen Version von EasyKit vor allem sehr einfache Mikrocontroller genutzt werden, welche zwar eher geringe Verarbeitungsgeschwindigkeiten besitzen, dafür aber weniger Bauraum benötigen, kommt auf dem Starterboard ein leistungsstarker Cortex™-M3 zum Einsatz. Um diesen programmieren zu können, muss das Board lediglich mittels eines USB-Kabels an einen PC mit installiertem EasyLab angeschlossen werden. [BA2010a]

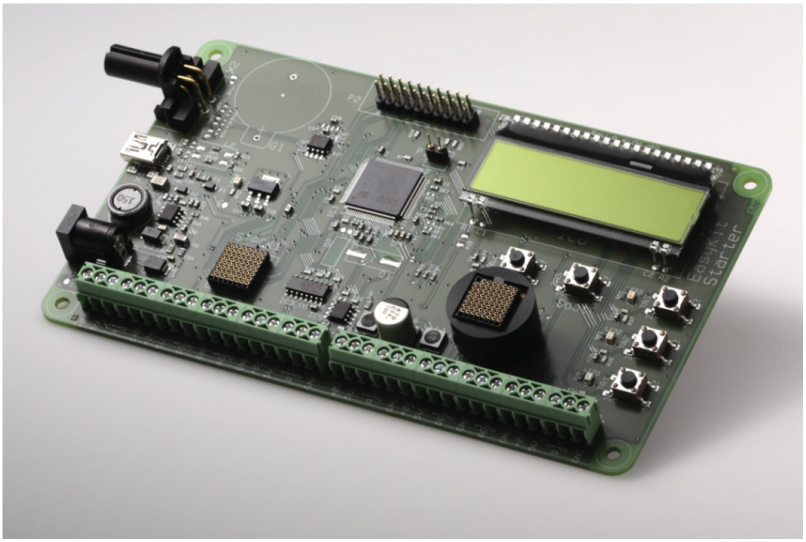

Abbildung 27: EasyKit Starterboard [Fes2010]

3.2.2 EasyKit Applikationsboard

Das neben dem Starterboard im EasyKit Starter enthaltene Applikationsboard ist in Abbildung 28 dargestellt. Es beinhaltet einen Gleichstrommotor, eine Gabellichtschranke und einen Lichtstärkesensor. Mit diesen Bestandteilen kann beispielsweise die Geschwindigkeit des Motors anhand der Lichtstärke oder anhand von in EasyLab vorgegebenen Sollwerten geregelt werden. Auf dem Board befinden sich alle benötigten Treiber- und Wandlerschaltungen zum Betrieb dieser Bauteile, so dass über die Schraubkontakte lediglich eine Verbindung zum Starterboard hergestellt werden muss. Um den EasyKit Starter an Schulen einsetzen zu können, war es nötig, einige Beispielanwendungen zu ermöglichen, welche den Schülern zeigen, für welche Aufgaben Mikrocontroller genutzt werden können. Beim Einsatz der industriellen Version von EasyKit liegt gewöhnlich bereits ein technisches Problem vor, welches gelöst werden soll. Da dies bei der Nutzung in der Lehre an Schulen nicht der Fall ist, wurde die Entwicklung des Applikationsboards notwendig. [BA2010a]

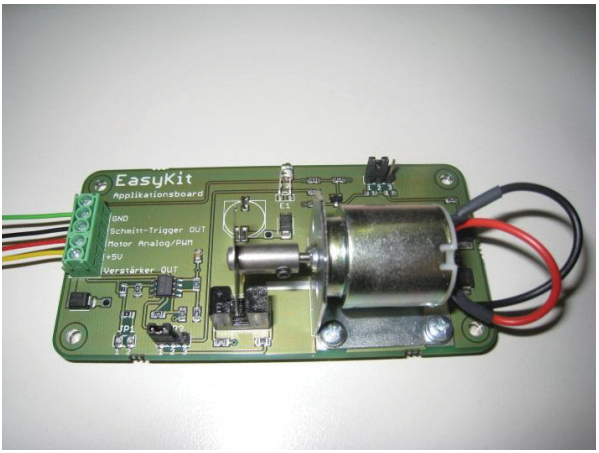

Abbildung 28: Applikationsboard

3.2.3 Vereinfachung von EasyLab für den EasyKit Starter

Die grundlegende Struktur der Entwicklungsumgebung EasyLab wurde für den EasyKit Starter nicht verändert. Die Programmierung erfolgt genau wie bei der industriellen Version von EasyKit mit Hilfe der Ablaufsprache und der Datenflusssprache. Auch die genutzten plattformübergreifenden Funktionsbausteine der Datenflusssprache sind in beiden Versionen identisch. Diese wurden zwar im Verlauf der Entwicklung des EasyKit Starters überarbeitet, jedoch wurden die verbesserten Funktionsbausteine in dieser Form auch in die industrielle Version von EasyLab übernommen.

Ein wichtiger Unterschied zur industriellen Version ist die Vereinfachung des Herunterladens des Programms auf den Mikrocontroller. Während bei der industriellen Version vom Nutzer verschiedene Einstellungen zur Auswahl und Parametrierung des genutzten Mikrocontrollers abgefragt werden, läuft beim EasyKit Starter der gesamte Vorgang mit den voreingestellten Parametern vollautomatisch ab. Dies vereinfacht die Bedienung erheblich und sorgt für weniger Missverständnisse. Erfahrene Nutzer können über Umwege dennoch eine Parametrierung vornehmen.

Ein weiterer wichtiger Unterschied ist die Bereitstellung einiger Beispielaufgaben mit Lösungen innerhalb von EasyLab. Die Beispielaufgaben kön-

nen selbstständig durchgeführt werden, um den Aufbau des Systems kennen zu lernen. Bis zum jetzigen Zeitpunkt stehen sechs Aufgaben zur Verfügung.

Da EasyLab bereits mit der Zielstellung der Nutzerfreundlichkeit entwickelt wurde, waren nur diese wenigen Schritte erforderlich, um die industrielle Version zu einer Schulversion umzuarbeiten.

3.2.4 Vor der Nutzung des EasyKit Starters

Wie in den letzten Abschnitten beschrieben wurde, ist der EasyKit Starter bereits vergleichsweise einfach nutzbar. Vor allem die Implementierung leicht verständlicher graphischer Programmiersprachen bewirkt diese einfache Nutzbarkeit. Dennoch sind für die Nutzung des Systems einige Grundlagenkenntnisse erforderlich. Daher wurde zur Unterstützung des Einstiegs in die Problematik der mikrocontrollerbasierten mechatronischen Systeme ein web-based Training entwickelt.

Hinter dem Namen web-based Training versteckt sich ein modernes Lehrprogramm, in welchem zu vermittelnde Lehrinhalte für ein selbstständiges Erlernen am PC aufbereitet wurden. Durch ein solches System ist es möglich, die Vorteile moderner Medien zu nutzen, um Wissen gezielt zu vermitteln. Der Aufbau und die Inhalte des web-based Trainings sind als Übersicht in Abbildung 29 zu finden.

Das web-based Training beginnt mit allgemeinen Informationen über dessen Nutzung. Das darauf folgende Einführungskapitel soll die Schüler motivieren, indem Informationen vermittelt werden, in welchen Bereichen im Alltag wir bereits mit Mikrocontrollern in Berührung kommen und welche Aufgaben diese dort übernehmen. Diese Informationen werden zunächst auf einer sehr anwendungsnahen Ebene vermittelt. Beispielsweise wird den Schülern die Aufgabe von Mikrocontrollern verständlich gemacht, indem die Funktionsweise eines Kaffeeautomaten erklärt wird. Dabei werden nicht die elektronischen Abläufe beschrieben, wie zum Beispiel das Schalten einzelner Mikrocontrollerpins, sondern eher Vorgänge auf einer höheren Ebene, wie zum Beispiel das Aufheizen oder das Pumpen von Wasser. Auf diese Art lassen sich beispielsweise grundlegende Begriffe, wie Sensor und Aktor, einführen.

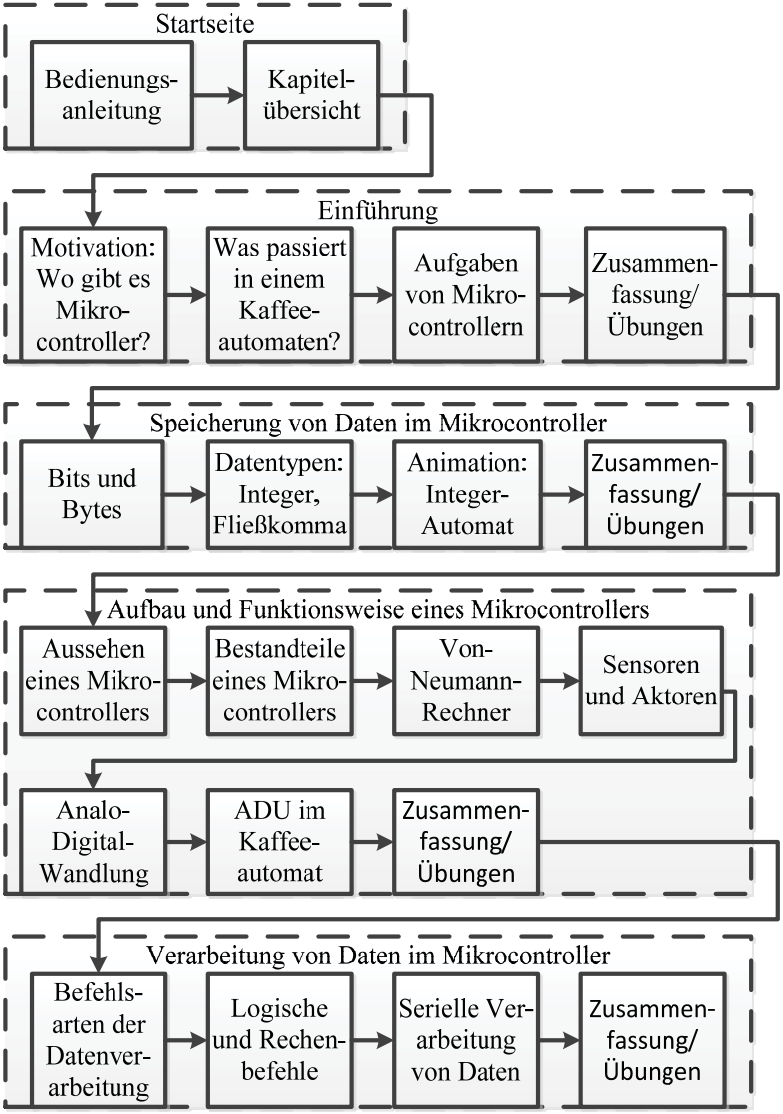

Abbildung 29: Inhalte des web-based Trainings

Im zweiten Kapitel erfahren die Schüler einige Informationen über die Speicherung von Daten innerhalb des Mikrocontrollers. Das Konzept der Nutzung von Bits und Bytes erklärt und die Darstellungsmöglichkeit von ganzen und gebrochenen Zahlen in Form von Bytes beschrieben. Außerdem werden einige gebräuchliche Datentypen erklärt. Dies ist notwendig, da die Schüler später in EasyLab mit diesen Datentypen arbeiten müssen.

Das dritte Kapitel beschreibt den prinzipiellen Aufbau von Mikrocontrollern anhand der von-Neumann-Architektur. Die Funktionen der einzelnen Bestandteile des Controllers sowie ihre Zusammenarbeit werden näher erklärt. Besonderes Augenmerk liegt dabei auf der Beschreibung der Ein- und Ausgangsmodule. Das Wissen über diese Module ist essentiell, um später mit dem EasyKit Starter und EasyLab arbeiten zu können, da beispielsweise die Funktionsweise der Verarbeitung analoger und digitaler Signale bekannt sein muss. Dabei wird immer wieder auf das leicht verständliche Beispiel des Kaffeeautomaten eingegangen, um den Schülern das Verständnis zu erleichtern.

Das letzte Kapitel beschäftigt sich mit verschiedenen Operationen des Mikrocontrollers, welche zur Verarbeitung von Daten genutzt werden. Während der Programmierung des Datenflussplans in EasyLab werden diese Operationen oft benötigt, so dass eine Einführung in dieses Thema notwendig ist.

Am Ende eines jeden Kapitels erfolgt eine Zusammenfassung. Anschließend werden dem Schüler zwei Fragen gestellt. Die Beantwortung dieser Fragen soll zum einen eine Überprüfung der eigenen Leistung für den Schüler sein, zum anderen soll sie das erlernte Wissen noch einmal festigen. Um EasyLab nutzen zu können, muss sich der Schüler jedoch nicht an alle Einzelheiten des web-based Trainings erinnern. Wichtig ist bei der Nutzung des Trainings, dass die Schüler einige Grundzusammenhänge verstanden haben, um die Funktionen des Systems nachvollziehen zu können.

3.2.5 Das EasyKit Gesamtkonzept

Der EasyKit Starter fügt sich nahtlos in das Gesamtkonzept von EasyKit ein. Absolute Neueinsteiger beginnen mit diesem System. Sie arbeiten sich zuerst durch das web-based Training und können durch die Bearbeitung der Einführungsaufgaben sehr schnell zu einem ersten Erfolgserlebnis gelangen.

Mit steigender Nutzungsdauer wird der Nutzer immer neue Aufgaben für das System finden, welche ab einer bestimmten Komplexität nicht mehr durch die Nutzung des Starterboards gelöst werden können. An diesem Punkt besteht die Möglichkeit, zur industriellen Version von EasyKit zu wechseln. Durch seine höhere Flexibilität können hiermit mehr Aufgaben gelöst werden. Der Vorteil an diesem Schritt ist, dass sich der Nutzer nicht in eine neue Programmierumgebung einarbeiten muss, da nur geringe Veränderungen zwischen dem EasyLab der industriellen und der Starterversion vorgenommen wurden.

Auch die industrielle Version von EasyKit wird ab einer bestimmten Aufgabenkomplexität nicht mehr ausreichend flexibel sein. Ist dabei der modulbasierte Aufbau von EasyLab der beschränkende Faktor, können die EasyKit-Elektronikmodule auch mit einer gewöhnlichen textbasierten Entwicklungsumgebung programmiert werden. Der Vorteil hier ist, dass der Nutzer sich nicht mit der Entwicklung der elektronischen Schaltungen beschäftigen muss, sondern sich ganz auf das Erlernen der Programmierung konzentrieren kann. So ist es möglich, unerfahrene Nutzer schrittweise an das Thema mikrocontrollerbasierter mechatronischer Systeme heranzuführen.

3.3 Nutzertests

Der EasyKit Starter wurde gemeinsam mit verschiedenen Zielgruppen getestet. Die Ergebnisse der Tests werden im Folgenden kurz dargestellt.

3.3.1 Expertentests

Der erste hier beschriebene Test des EasyKit Starters fand im Rahmen der Abschlussveranstaltung des EasyKit-Projekts statt. Die etwa 40 anwesenden Vertreter aus Industrie und Forschung erhielten zunächst eine grundlegende Einweisung in die Hardware des Systems. Dabei wurde auf die Nutzung des web-based Trainings verzichtet, da das hierin enthaltene Grundlagenwissen hinlänglich bekannt war. Anschließend lösten sie unter Anweisung des Vortragenden eine Beispielaufgabe, bei welcher die drei auf dem Starterboard befindlichen LEDs (lichtemittierende Dioden) mittels der ebenfalls auf dem Board befindlichen Taster geschaltet werden sollten. Danach sollte das Pro-

gramm selbstständig erweitert werden, so dass die LEDs beim Drücken eines Tasters einschalten und nach einer vorgegebenen Zeit wieder ausschalten. Nachdem alle Testnutzer dieses Problem gelöst hatten, wurde eine weitere Beispielaufgabe unter Anweisung des Vortragenden gelöst. Im Rahmen dieser Aufgabe sollte das Signal der Lichtschranke des Applikationsboards ausgelesen werden. Auch dieses Programm sollte im Anschluss selbstständig erweitert werden, um die Motordrehzahl des Applikationsboards regeln zu können. Abschließend erhielten die Probanden die Möglichkeit, selbstständig weitere Funktionen des Systems zu testen.

Da bei der Veranstaltung nicht der Test des Systems im Vordergrund stand, sondern die Präsentation des Projekts, wurde auf eine intensive Befragung der Probanden verzichtet. Jedoch wurden diese während der Nutzung beobachtet und auftretende Probleme dokumentiert.

Die Nutzer waren grundsätzlich mit dem System zufrieden und konnten die gestellten Aufgaben lösen. Lediglich bei der Erweiterung der ersten Beispielaufgabe traten einige Schwierigkeiten auf. Der Grund hierfür lag darin, dass für die Lösung der Beispielaufgabe lediglich die Programmierung in der Funktionsbausteinsprache genutzt wurde, während sich die Erweiterung sehr einfach in der Ablaufsprache umsetzen ließ. EasyKit startet standardmäßig mit der Anzeige der Oberfläche, in welcher mit der Funktionsbausteinsprache programmiert wird. Die Oberfläche, in welcher mit der Ablaufsprache gearbeitet wird, ist in einem separaten Reiter zu finden, welcher auf den ersten Blick nicht auffällt. Außerdem wurde die Programmierung in der Ablaufsprache während des Lösens der ersten Beispielaufgabe nicht thematisiert. Daher versuchten die Teilnehmer, das Problem in der Funktionsbausteinsprache zu lösen, indem beispielsweise Zählschleifen oder Flip-Flops genutzt wurden. Die entwickelten Lösungen waren somit zwar sehr kreativ, jedoch auch äußerst ineffizient. Alles in allem konnten Nutzer mit dem System aber sehr gut arbeiten und die jeweils gewünschte Funktionalität realisieren.

3.3.2 Tests mit Lehrern

Nach der Entwicklung des EasyKit Starters wurden etwa 300 Stück davon an Hauptschulen, Realschulen, Gymnasien und Berufsschulen verlost. Für

die Lehrer an diesen Schulen wurden Schulungen (Workshops) angeboten, um das System kennen zu lernen.

Durchführung

Bei den Workshops erhielten die Lehrer eine Einweisung in das System und führten praktische Übungen durch, in welchen Aufgaben mit und ohne Anleitung gelöst werden mussten. Zum Zweck der Evaluierung wurden am Ende der Veranstaltung von insgesamt 83 Teilnehmern Fragebögen ausgefüllt. Darin wurden zunächst Informationen über die jeweilige Schulart und die unterrichteten Schulfächer erfragt. Außerdem sollte angegeben werden, ob an der Schule bereits Mikrocontroller genutzt werden und ob damit verbundene Themen im Unterricht behandelt werden. Im Anschluss sollten die Lehrer die graphisch orientierte Programmierung und den Simulationsmodus nach seiner Vorteilhaftigkeit beurteilen und dies nach Möglichkeit begründen. Abschließend wurden die Lehrer gebeten, sich zur Durchführung des Workshops zu äußern und weitere Anmerkungen und Anregungen zu geben.

Ergebnisse

Das Feld der Teilnehmer der Schulungsveranstaltung setzte sich aus 31 Lehrern von Gymnasien, 21 Lehrern von Berufsschulen und 31 Lehrern von Haupt- und Realschulen zusammen. Allerdings ist hierbei zu erwähnen, dass sich vor allem technisch orientierte Schulen mit fachlich interessierten Lehrern beworben hatten, so dass die im Folgenden genannten Zahlen nur für solche Schulen zutreffend sind. Vor allem im Bereich der behandelten Themengebiete sind an nicht technisch ausgerichteten Schulen bedeutend geringere Zahlen zu erwarten. In Abbildung 30 ist dargestellt, welche mikrocontrollerbezogenen Themengebiete an den jeweiligen Schulen behandelt werden.

Dabei ist vor allem auffällig, dass lediglich 43% der Lehrer der Gymnasien angeben, dass an ihren Schulen bereits Mikrocontroller in der Ausbildung genutzt werden. Bei den Berufsschulen sind es hingegen bereits 75%. Das bedeutet, dass viele zukünftige Studenten während ihrer Schulzeit noch keinen Kontakt zu Mikrocontrollern hatten, so dass sie sich während ihres Studiums von Grund auf in dieses Thema einarbeiten müssen.

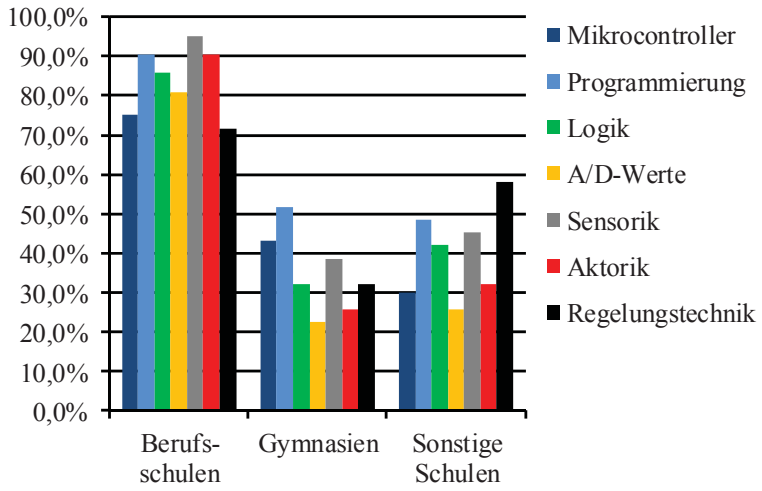

Abbildung 30: Mikrocontrollerbezogene Themengebiete, welche an verschiedenen Schularten unterrichtet werden [BA2010a]

Auf die Fragen, ob die Nutzung der graphisch orientierten Programmierung und des Simulationsmodus vorteilhaft im Unterricht sei, antwortete lediglich ein Lehrer negativ. Besonders bemerkt wurde, dass durch die Nutzung des Simulationsmodus nicht unbedingt jeder Schüler ein Starterboard zur Verfügung haben muss, um die Software zu entwickeln. Außerdem erkannten viele Lehrer, dass aufgrund der graphisch orientierten Programmierung ein anschauliches und einfaches Arbeiten mit Mikrocontrollern möglich ist, so dass die Schüler die grundsätzliche Struktur und Funktion mikrocontrollerbasierter mechatronischer Systeme, aber auch angrenzende Wissensfelder, wie die Regelungstechnik, leichter verstehen können, ohne zuvor die theoretisches Grundlagen dieser Fachgebiete erlernt haben zu müssen.

Die einfache Nutzbarkeit des Systems wurde also von den Lehrern als positiv bewertet. Als Anregungen gaben mehrere Teilnehmer der Schulung jedoch zu bedenken, dass momentan zu wenige Anwendungsmöglichkeiten zur Verfügung stehen, so dass an dieser Stelle noch einige Erweiterungen zu durchdenken sind.

3.3.3 Nutzertests mit Schülern

Nachdem der EasyKit Starter zunächst von Experten aus Industrie und Forschung sowie von Lehrern getestet wurde, erfolgten weitere Tests mit Schülern, welche die eigentliche Zielgruppe des Systems darstellen. Dabei standen zwei Aspekte im Fokus:

- Zum einen sollte getestet werden, ob die Nutzer den EasyKit Starter mit nur wenig oder ganz und gar ohne domänenspezifisches Expertenwissen nutzen können. Für den Fall, dass jedoch Expertenwissen erforderlich sei, sollte dieses festgehalten werden.
- Zum anderen wurde untersucht, welches Wissen die Schüler während der Nutzung des Systems über den Aufbau, die Funktion und die Aufgaben mikrocontrollerbasierter mechatronischer Systeme sowie über deren Programmierung erlernen. Hierzu gehören beispielsweise der Aufbau unterschiedlicher Signalarten oder Grundlagen der Programmierung, wobei unter dem zweiten Punkt nicht das Erlernen einer Programmiersprache zu verstehen ist, sondern lediglich das Verständnis einiger Programmierkonzepte und Strukturen, wie beispielsweise Schleifen, Verzweigungen oder die Existenz spezifischer Datentypen.

Ablauf des Nutzertests

Für die Durchführung der Nutzertests konnten 49 Jungen und 50 Mädchen im Alter von 14 und 15 Jahren an einem Gymnasium gewonnen werden. Jeweils drei bis vier Schüler arbeiteten gemeinsam in einer Gruppe an einem System. Die Tests erfolgten im Klassenverband und wurden von drei älteren Schülern der Schule begleitet. Diese hatten sich bereits vorher intensiv mit dem System beschäftigt und unterstützten bei der Durchführung der Tests genauso wie bei der Bearbeitung der Fragebögen.

Da das web-based Training theoretische Grundlagen vermittelt, der Nutzertest aber mit Nutzern ohne Vorwissen durchgeführt werden sollte, kam das web-based Training im Rahmen dieses Tests nicht zum Einsatz. Als motivierende Einführung wurde den Schülern mit einfachen Beispielen vermittelt, was sich hinter den Begriffen der eingebetteten und der mechatronischen Systeme verbirgt und wie sie im täglichen Leben mit diesen Systemen

in Berührung kommen. Dem schloss sich das vom Testleiter geführte Lösen der in EasyLab integrierten Beispielaufgaben an. Nach jeder gelösten Aufgabe, sollten die Schüler die jeweilige Anwendung erweitern. Die Bandbreite der Aufgabenstellungen erstreckte sich von einem einfachen „Hello World"-Programm bis hin zur Drehzahlregelung des Motors des Applikationsboards.

Die Evaluierung erfolgte in Form eines Usability Tests, wie er in Abschnitt 2.3.5 vorgestellt wurde. Dabei beobachteten die Tester die Nutzer, während diese die Aufgabenstellungen lösten. Beim Auftreten von Problemen wurden sie darum gebeten, das Problem und die damit verbundenen Bestandteile der Oberfläche zu erklären, so dass eine vereinfachte Form der „Thinking Aloud"-Methode zu Einsatz kam. Die Erklärung der Bestandteile der Oberfläche war notwendig, um zu erkennen, ob die Nutzer bestimmte Dialoge so verstanden, wie dies von den Entwicklern vorgesehen war. Nach Abschluss des Tests füllten die Schüler Fragebögen aus und einige Schüler wurden in Form informeller Interviews zu einzelnen, ausgewählten Problemen befragt. Die Fragebögen sollten vor allem aufdecken, welche Bestandteile des EasyKit Starters die meisten Probleme verursachten. Diese Probleme konnten im Verlauf der persönlichen Befragungen genauer spezifiziert werden und Lösungsansätze hierfür konnten gemeinsam mit den Schülern erarbeitet werden. Außerdem sollte in der Befragung geklärt werden, welche Aspekte des Themengebietes der mechatronischen Systeme die Schüler durch die Nutzung des Systems verstanden hatten.

Testergebnisse

Sämtliche Teilnehmer des Tests gaben an, dass sie sich vorher noch nie mit mikrocontrollerbasierten mechatronischen Systemen beschäftigt hätten. Allerdings hatten bereits 32% der Schüler versucht, Software für PCs zu programmieren.

Ein wichtiges Ergebnis war, dass sich etwa 20% der Schüler im Fragebogen gegen einen Einsatz im Unterricht aussprachen, wobei etwa die Hälfte dieser Schüler angab, dass sie sich keinen weiteren Verwendungszweck vorstellen könnten. Da dies bereits genauso von den Lehrern bemängelt wurde und zudem den Einsatz des EasyKit Starters im Hobbybereich deutlich erschweren

würde, muss dieser Aspekt bei zukünftigen Versionen berücksichtigt werden.

Grundsätzlich kann aber gesagt werden, dass das System von den Schülern gut akzeptiert wurde. Im Fragebogen erklärten 83% der Schüler, dass die Arbeit mit dem System für sie interessant war, obwohl 31% angaben, dass sie im Verlauf des Nutzertests Hilfe benötigten.

Die Auswertung der Fragebögen zeigte, dass annähernd die Hälfte der Probleme auftrat, als während der Programmierung im Datenflussplan versucht wurde, Funktionsbausteine miteinander zu verbinden. Dieses Problem tritt auf, wenn der Datentyp des Ausgangs eines Funktionsbausteins nicht mit dem Datentyp des zu verbindenden Eingangs des folgenden Funktionsbausteins kompatibel ist. Zwar können, genau wie bei der textbasierten Programmierung, einige Datentypen ineinander überführt werden, bei allen ist dies allerdings nicht möglich. Beispielsweise ist es problemlos möglich, eine nicht vorzeichenbehaftete Integervariable mit 16bit Auflösung in eine andere nicht vorzeichenbehaftete Integervariable mit 32bit Auflösung zu speichern. Die entgegengesetzte Richtung ist nur unter bestimmten Bedingungen möglich und kann schnell zu Fehlern in der Berechnung führen. Auch das Speichern vorzeichenbehafteter Variablen in nichtvorzeichenbehaftete Variablen stellt ein Problem dar. Daher ist dies in EasyLab nicht möglich beziehungsweise es muss ein spezieller Funktionsbaustein zur Datentypumwandlung genutzt werden. Das in EasyLab gewünschte Vorgehen ist allerdings, dass sich der Nutzer während der Programmierung selbstständig Gedanken über die Datentypauswahl macht und diese Auswahl konsistent umsetzt. Hierfür ist es allerdings unerlässlich, dass er die Eigenschaften der genutzten Datentypen kennt.

Im Rahmen des Interviews erklärten die Schüler, dass sie vor allem an zwei Stellen scheiterten. Zunächst war es für viele Schüler gar nicht offensichtlich, dass der Datentyp unter bestimmten Umständen angepasst werden musste, aber auch wenn die Schüler dieses Problem erkannten, wussten sie nicht, welchen Datentyp sie wählen sollten.

Hierfür gibt es mehrere Gründe:

- Natürlich ist es sinnvoll, dass inkompatible Ein- und Ausgänge nicht verbunden werden können. Allerdings erhält der Nutzer keine Rückmeldung, warum die Verbindung nicht möglich ist. Eine aussagekräftige Fehlermeldung wäre an diesem Punkt wünschenswert. Außerdem ist der aktuell gewählte Datentyp nur zu erkennen, wenn der Nutzer den Dialog zur Parametrierung des Funktionsbausteins durch einen Doppelklick öffnet.
- EasyLab ermöglicht die Arbeit auf den drei Komplexitätsstufen einfach, mittel und fortgeschritten. Je niedriger die Stufe, desto mehr komplexe Details werden ausgeblendet. Die Datentypauswahl steht nur im Fortgeschrittenenmodus zur Verfügung, so dass bei der Arbeit im einfachen oder mittelschweren Modus gar keine Möglichkeit besteht, die Datentypen anzupassen. Für einige Aufgaben ist dies jedoch erforderlich.
- Die Bezeichnungen der Datentypen in EasyLab basieren auf Standardtypen, weshalb sie für Programmierer leicht zu verstehen sind. Da Bezeichnungen wie uint16 aber nicht im natürlichen Sprachgebrauch genutzt werden, sind sie für Einsteiger ohne Grundlagenwissen unverständlich.

Ein weiteres Problem zeigte sich, als die Schüler selbstständig ein Programm so erweitern sollten, dass dieses zeitlich gesteuerte Abläufe ermöglicht. Genau wie die in Abschnitt 3.3.1 vorgestellten Testnutzer aus Industrie und Forschung, wollten die Schüler den zeitlichen Ablauf mit Hilfe von Zählschleifen in der Funktionsbausteinsprache realisieren, obwohl die Programmierung in Ablaufsprache hierfür besser geeignet gewesen wäre. Dieses Problem trat bei etwa der Hälfte der Gruppen auf. Es zeigte sich, dass viele dieser Gruppen gar nicht realisiert hatten, dass das in der Funktionsbausteinsprache programmierte Datenflussprogramm eigentlich nur ein Teil des Gesamtprogramms ist. Der Grund hierfür ist, dass nach dem Start von EasyLab nicht das Ablaufdiagramm im Vordergrund dargestellt wird, sondern ein Datenflussplan mit dem Namen „Step1“. Das Ablaufdiagramm befindet sich in einem zweiten Reiter im Hintergrund. Es besteht, wie in Ab-

bildung 31 dargestellt, zu Beginn lediglich aus einem Initialisierungszustand, welcher keine Anweisungen enthält und einem weiteren Zustand, welcher das Datenflussprogramm „Step1“ beinhaltet. Wie an den Bedingungen an den gerichteten Verbindern in Abbildung 31 zu erkennen ist, wird der Initialisierungszustand einmal durchlaufen und danach direkt zum nächsten Zustand übergegangen. Dieser wird immer wieder wiederholt.

Durch die Vorgabe dieses Ablaufdiagramms kann der Nutzer ein vollständig funktionierendes Programm erzeugen, indem er ausschließlich mit der Funktionsbausteinsprache im Datenflussplan arbeitet. Für einfache Regelungs- und Steuerungsaufgaben ohne ereignisgesteuerte Abläufe ist dies ausreichend. In anderen Fällen muss auch das Ablaufdiagramm erweitert werden. Dieser Aufbau ist für den Nutzer zunächst von Vorteil, da er sich beim Lösen einfacher Aufgaben nicht um die Bearbeitung des Ablaufdiagramms kümmern muss. Allerdings tritt dieses in den Hintergrund und kann in dem zweiten Reiter leicht übersehen werden. Des Weiteren beschäftigt sich keine der sechs Beispielaufgaben mit der Entwicklung eines ereignisgesteuerten Ablaufdiagramms. Folglich wird während der Bearbeitung der Beispielaufgaben nicht mit der Ablaufsprache gearbeitet.

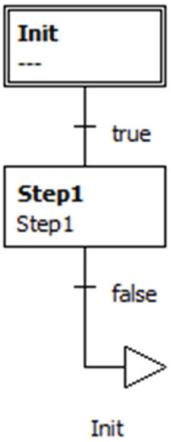

Abbildung 31: Standardablaufdiagramm des Hauptprogramms in EasyLab

Doch auch bei Schülern, welche das Ablaufdiagramm zur Programmierung nutzten, zeigten sich einige Schwierigkeiten. So vermuteten einige Teilnehmer zunächst, dass die einzelnen Zustände jeweils nur einmal durchlaufen werden und das Programm am Ende einfach anhält. Erst während der Simulation des erzeugten Programms erkannten diese Schüler die eigentliche Funktionsweise.

Des Weiteren wurde durch die Beobachtung der Schüler festgestellt, dass die Art der Erstellung des Ablaufdiagramms nicht sehr intuitiv ist. Grund hierfür ist, dass sich die Darstellung des Diagramms an der DIN 61131 [DIN61131-3-2003] orientiert. Dadurch benötigten die Testnutzer mehrere Versuche, bis sie das gewünschte Ablaufdiagramm erstellt hatten.

Neben der Spezifizierung der aufgetretenen Probleme sollte im Rahmen der Interviews auch untersucht werden, welches Wissen sich die Schüler während der Nutzung des EasyKit Starters aneigneten. Hierfür wurden den Schülern verschiedene Fragen über den Aufbau und die Funktionsweise der Hard- und Software des EasyKit Starters und allgemein von mechatronischen Systemen gestellt. Die Befragung erfolgte in Form eines freien Interviews, für welches vorher keine speziellen Fragen festgelegt wurden. Es zeigte sich, dass die meisten Schüler die wichtigsten Aspekte der Programmierung verstanden hatten. Auch wenn sie die exakten Begriffe hierfür nicht nannten, so erkannten sie dennoch zum einen die Existenz von Schleifen und Verzweigungen, zum anderen aber auch die typischen Aufgabenstellungen von mikrocontrollerbasierten mechatronischen Systemen. Hardwarebezogene Aspekte wurden von den Schülern allerdings nur in geringem Maße verstanden. Vor allem die Existenz und Verarbeitung unterschiedlicher Signalarten sowie die Problematik unterschiedlicher Spannungsniveaus und Stromstärken erkannten sie nicht. So dachten beispielsweise einige Schüler, dass sie einen Gleichstrommotor direkt an einen analogen Ausgang des Starterboards anschließen könnten. Solche grundlegenden hardwarebezogenen Probleme werden in EasyLab nicht betrachtet. Die Repräsentation der Hardware beschränkt sich auf die wichtigsten Funktionen der Mikrocontrollerpins. Dies beinhaltet die Nutzung dieser Pins als analoge Eingänge, als digitale Ein- und Ausgänge oder als Flankenzähler. Ein Bezug zur ange-

schlossenen spezifischen Hardware, wie zum Beispiel Motoren oder Sensoren, wird nicht hergestellt.

Zusammenfassung der erkannten Probleme

Im Folgenden werden die in den Nutzertest erkannten Probleme des EasyKit Starters noch einmal in Kurzform aufgezählt, da diese als Grundlage für die Weiterentwicklung dienten:

- Aus Sicht der Lehrer und der Schüler ist der praktische Einsatzbereich beschränkt, so dass eine weitere Nutzung im Hobbybereich eher unwahrscheinlich ist.
- Die Nutzer erkannten nicht, dass für ereignisgesteuerte Vorgänge die Ablaufsprache vorgesehen ist, da diese in den Hintergrund tritt, obwohl sie die Struktur des Hauptprogramms bestimmt.
- Außerdem hatten sie Schwierigkeiten, die Funktionsweise des Ablaufdiagramms richtig zu verstehen und dieses zu verändern.
- Die Schüler trafen auf Probleme, da sie die genutzten Datentypen anpassen mussten. Aus der Software war jedoch nicht ersichtlich, dass und wie dies zu erfolgen hat.
- Hardwarebezogene Aspekte der Entwicklung mikrocontrollerbasierter mechatronischer Systeme wurden nur in geringem Umfang verstanden, da diese innerhalb der Software nicht thematisiert werden.

3.4 Zusammenfassung

Dieses Kapitel stellte zunächst die industrielle Version von EasyKit vor. Es wurde aufgezeigt, wie ein mechatronisches System mit der modularen Hardware und der Software EasyLab entwickelt und programmiert werden kann. Anschließend erfolgte die Beschreibung des EasyKit Starters, welcher Schülern das Thema der mechatronischen Systeme näherbringen kann. Dieser Abschnitt beinhaltete auch die Vorstellung der Anpassungen von EasyLab zur besseren Nutzbarkeit sowie die Beschreibung des neu erstellten

web-based Trainings. Als Letztes wurden für den EasyKit Starter Ergebnisse von Nutzertests mit Experten, Lehrern und Schülern aufgeführt, wodurch verschiedene Defizite des bestehenden Systems aufgezeigt werden konnten.

Es kann festgehalten werden, dass die beiden Systeme für die vorgesehenen Zwecke gut geeignet sind. Bei der industriellen Version von EasyKit kann mit den Abschlägen in Sachen Nutzerfreundlichkeit gut umgegangen werden, da dieses System in erster Linie Entwicklungszeit sparen soll und vor allem von Nutzern mit Expertenwissen eingesetzt wird. Der EasyKit Starter kann in Verbindung mit dem web-based Training und einer fachkundigen Anleitung durch einen Lehrer auch sehr gut im Schulunterricht eingesetzt werden. Verschiedene Ansätze beider Systeme bilden daher eine gute Grundlage für die Weiterentwicklung zu einem flexiblen und plattformunabhängigen System, welches auch ohne Expertenwissen genutzt werden kann und somit auch im Hobbybereich einsetzbar ist. Die Ergebnisse der durchgeführten Nutzertests sollen hierbei helfen.

4 Erweiterte Entwicklungskonzepte

Auf Basis der in Abschnitt 3.3 beschriebenen Testergebnisse des EasyKit Starters wurden neue Konzepte entwickelt, welche die Entwicklung mikrocontrollerbasierter mechatronischer Systeme im Hobbybereich ermöglichen. Dabei steht vor allem eine Unterstützung des Nutzers im Mittelpunkt, so dass dieser bei der Entwicklung kein domänenspezifisches Expertenwissen benötigt. Des Weiteren berücksichtigen die Konzepte die bereits angesprochenen Rahmenbedingungen der Plattformunabhängigkeit und der möglichst hohen Flexibilität bei der Programmierung. In diesem Kapitel werden Anforderungen hierfür vorgestellt und entsprechende Umsetzungskonzepte vorgeschlagen.

4.1 Drei-Ebenen-Konzept der Softwaremodellierung

In diesem Abschnitt wird die Möglichkeit diskutiert, Software auf unterschiedlichen Ebenen zu modellieren. Dabei wird auf der ersten Ebene die grundsätzliche Ablaufstruktur der Software festgelegt. Auf der zweiten Ebene wird die Funktionalität einzelner Elemente der ersten Ebene modelliert. Auf der dritten Ebene können neue Elemente für die zweite Ebene definiert werden. Im Folgenden werden die Ebenen und die Motivation für ihre Einführung vorgestellt. Dabei wird zunächst von Beispielsystemen ausgegangen, in welchen die Softwaremodellierung auf nur einer Ebene stattfindet. Schrittweise wird anschließend die Einführung der neuen Ebenen motiviert. Um das Verständnis der Anordnung der Ebenen und ihrer Beziehungen untereinander für den Leser zu erleichtern, ist in Abbildung 32 eine Überblicksdarstellung zu finden.

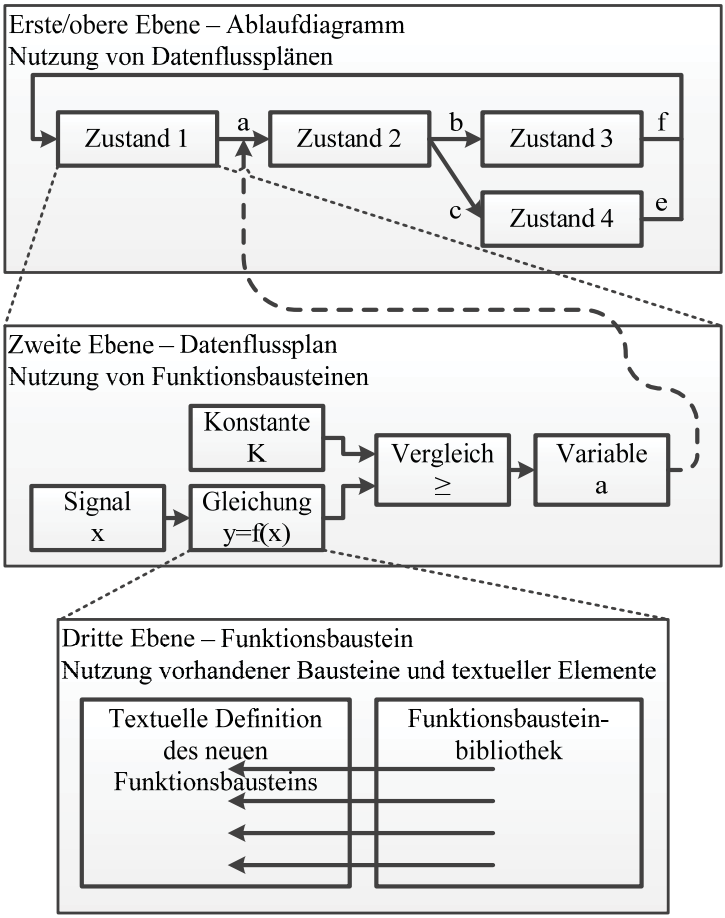

Abbildung 32: Darstellung der drei Ebenen der Programmierung und ihrer Abhängigkeiten

In mehreren der unter Abschnitt 2.4 vorgestellten Entwicklungsumgebungen wird die Software für das mechatronische System allein mit Hilfe eines Ablaufdiagramms erstellt, wobei einzelne Softwareblöcke, wie in Abbildung 9 dargestellt, hintereinander angeordnet werden. Diese Blöcke können zwar noch parametriert werden, ihre Grundfunktion ist jedoch stets die gleiche.

So gibt es in LEGO MINDSTORMS NXT beispielsweise einen Programmierblock, welcher zur Ansteuerung eines Motors genutzt wird. Soll sich der Motor drehen, so wird der Block in die Schrittfolge an die passende Stelle eingesetzt. Anschließend wird das Verhalten des Motors über Parameter festgelegt. Hierzu gehören zum Beispiel die Drehrichtung und die Drehdauer. Wird eine solche Ein-Ebenen-Darstellung genutzt, müssen sämtliche denkbaren Funktionen des Gesamtsystems bereits in den vorgefertigten Programmierblöcken implementiert sein. Existiert eine vom Nutzer gewünschte Funktion nicht, so muss diese von einem Experten implementiert werden oder bestimmte Aufgaben können nicht erfüllt werden. Wird Software für eine kontextspezifische Plattform, beispielsweise aus dem Bereich der Robotik, entwickelt, so kann eine solche Programmierung sinnvoll sein. Für eine flexible und plattformunabhängige Entwicklung ist diese jedoch nicht geeignet.

In EasyLab erfolgt daher die Programmierung in zwei hierarchischen Ebenen. Die obere, erste Ebene ist dabei ereignisgesteuert, während die zweite Ebene datenflussorientiert ist. Wie bereits in Abschnitt 3.1.2 beschrieben, werden in der oberen Ebene Zustände festgelegt, welche durch gerichtete Verbinder miteinander verkoppelt werden. Jedem der Zustände wird ein Datenflussplan zugewiesen, dessen Funktion auf der zweiten Ebene modelliert wird. Damit besitzen sie die gleiche Struktur, wie die in Abbildung 32 dargestellten oberen beiden Ebenen. Die Programmierung mit diesen beiden Ebenen hat sich in den Nutzertests weitestgehend bewährt, weist aber noch verschiedene Schwachpunkte auf. Wie in den Abschnitten 3.3.1 und 3.3.3 beschrieben, nutzten sowohl Experten als auch Schüler primär den Datenflussplan für die Programmierung, auch wenn eine Umsetzung im Ablaufdiagramm einfacher gewesen wäre. Hier liegt die Annahme nahe, dass der Datenflussplan für die Testnutzer einfacher verständlich war als das Ablaufdiagramm. Jedoch zeigen Erfahrungen mit anderen Entwicklungsumgebungen, wie zum Beispiel LEGO MINDSTORMS NXT, dass die Ablaufsprache auch von jungen Schülern gut verstanden wird. Aus diesem Grund und weil die Programmierung ereignisgesteuerter Vorgänge durch die Nutzung der Ablaufsprache deutlich erleichtert wird, ist es sinnvoll, den grundlegenden strukturellen Aufbau des Mikrocontrollerprogramms in einem Ablaufdiagramm zu modellieren. Um dies für den Nutzer zu vereinfachen, wurden

andere Darstellungsformen untersucht, welche besser verständlich sind und eine dialogbasierte Unterstützung während der Erstellung des Diagramms ermöglichen. Die Ergebnisse dieser Untersuchung sind in Abschnitt 4.2 beschrieben.

Auch die Nutzung von Datenflussplänen in der zweiten Ebene der Programmierung erwies sich als sinnvoll. Die Testnutzer verstanden die Funktionsweise sehr schnell. Zudem ermöglicht die Kombination der ereignisgesteuerten Modellierung der ersten Ebene mit der datenflussorientierten Modellierung dieser zweiten Ebene eine flexiblere Softwareentwicklung im Vergleich zur rein ereignisgesteuerten Programmierung. So können beispielsweise mathematische Berechnungen innerhalb des Datenflussplans deutlich einfacher erstellt werden. Die mögliche Funktionalität eines zu entwickelnden mechatronischen Systems wird durch den Umfang der Funktionsbausteinbibliothek eingeschränkt. Sind benötigte Funktionen nicht in Form eines Funktionsbausteins vorhanden, so wird ein Softwareentwickler mit domänenspezifischem Expertenwissen benötigt, um dies zu implementieren. Während der Entwicklung von Beispielanwendungen für EasyKit zeigte sich, dass regelmäßig neue Problemstellungen auftauchten, welche mit den vorhandenen Funktionsbausteinen nicht gelöst werden konnten. So wurde beispielsweise für eine Anwendung die Berechnung eines gleitenden Mittelwertes benötigt. Grundsätzlich war die Berechnung eines Mittelwertes auch vorher problemlos mit Hilfe von Additionen und einer Division möglich. Allerdings müssen beim gleitenden Mittelwert Daten von einem Durchlauf in den folgenden übernommen werden. Außerdem muss berücksichtigt werden, dass beispielsweise im ersten Durchlauf nur ein Wert, im zweiten Durchlauf zwei Werte usw. existieren. Allerdings stand zu diesem Zeitpunkt keine Möglichkeit zur Zwischenspeicherung eines Datums für einen späteren Durchlauf zur Verfügung. Daher musste ein neuer Funktionsbaustein erstellt werden.

Um dem Nutzer ein vergleichsweise einfach nutzbares Werkzeug in die Hand zu geben, mit welchem er selbst neue Funktionsbausteine für den Datenflussplan entwickeln kann, wurde eine dritte Ebene der Softwareentwicklung konzipiert. Im Folgenden werden hierfür die wichtigsten Anforderungen sowie mögliche Umsetzungskonzepte vorgestellt und diskutiert. Dabei werden zunächst die Ansätze für die Gestaltung der dritten Modellierungs-

ebene vorgestellt und näher erläutert. Anschließend wird diskutiert, welche Anforderung sich hieraus an die einzelnen Bestandteile des gewählten Modellierungsansatzes ergeben und mit welchen Umsetzungskonzepten diese erfüllt werden können.

Anforderungen und mögliche Konzepte der dritten Modellierungsebene

Die zentralen Anforderungen an die dritte Ebene sind:

- Eine hohe Flexibilität in der Programmierung wird vorausgesetzt, da die Ebene eingeführt wurde, weil die bisherige Flexibilität zu gering war.
- Auch Nutzer ohne Expertenwissen sollen Funktionsbausteine entwickeln können. Vor allem soll kein Wissen über eine textuelle Programmiersprache benötigt werden.

Auf der neu eingeführten dritten Ebene ist die Nutzung eines ablauforientierten Ansatzes sinnvoll. Das bedeutet, dass die Befehle schrittweise nacheinander ausgeführt werden, wobei auch Schleifen und bedingte Anweisungen im Ablauf vorkommen können. Unerfahrene Nutzer werden diese Ebene zunächst nicht benötigen, da die meisten typischen Problemstellungen mit den standardmäßig vorgegebenen Funktionsbausteinen gelöst werden können. Mit steigender Erfahrung wird der Nutzer jedoch auch auf weiterreichende Problemstellungen treffen, welche mit den vorhandenen Funktionsbausteinen entweder nur schwer oder gar nicht realisierbar sind. Die Anwender der dritten Programmierebene sind demnach meist Nutzer, welche bereits Erfahrung mit der Programmierung auf den beiden oberen Ebenen gesammelt haben, so dass ein gewisses Grundlagenwissen vorausgesetzt werden kann. Grundsätzlich kann die Entwicklung entweder in einer graphischen oder in einer textuellen Form erfolgen. Wird eine rein graphische Form gewählt, so ist eine möglichst einfache Darstellung eines Ablaufdiagramms sinnvoll, wie es in beispielhafter Form in Abbildung 33 dargestellt ist. Dabei werden die einzelnen Befehle in Form einer Kette nacheinander angeordnet. Anschließend werden diese parametriert, indem beispielsweise Bedingungen für die Verzweigungen eingegeben werden oder die Anzahl der Wiederholungen von Schleifen festgelegt wird. Der graphische Ansatz ist relativ einfach zu bedienen.

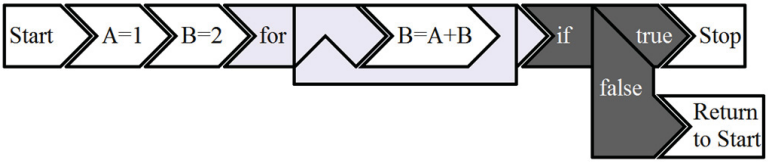

Abbildung 33: Ablaufdiagramm zur Erstellung eines benutzerdefinierten Funktionsbausteins mit einer Schleife und einer Verzweigung

Allerdings hat dieser den Nachteil, dass die mögliche Funktionalität des neuen Funktionsbausteins auf Kombinationen der vorhandenen Bausteine beschränkt ist.

Wird eine rein textuelle Form der Darstellung gewählt, so muss der Nutzer den Quellcode selbst eingeben. Dies ist vergleichbar mit der Softwareentwicklung innerhalb einer textbasierten Entwicklungsumgebung. Im Vergleich zu einer graphischen ist eine rein textuelle Darstellung deutlich schwieriger zu verstehen, da der Nutzer eine textbasierte Programmiersprache erlernen muss. Auch wenn diese Herangehensweise eine hohe Flexibilität ermöglicht, ist sie im Rahmen eines Systems für Einsteiger nicht praktikabel.

Um die Vorteile sowohl der graphischen als auch der textuellen Darstellungsform zu nutzen, wurde ein hybrides Konzept entwickelt, bei welchem Elemente beider Formen kombiniert sind. Der Quellcode des neu zu definierenden Funktionsbausteins wird dabei in textueller Form dargestellt, während die vorgegebenen Funktionsbausteine, aus welchen der neue Quellcode zusammengestellt wird, in graphischer Form vorliegen.

Als Sprache kann beispielsweise C genutzt werden. Dies hat verschiedene Vorteile:

- C kann auch von Nutzern ohne oder mit nur wenig Vorwissen vergleichsweise einfach verstanden werden, da es nur wenige Schlüsselwörter besitzt, welche sich weitestgehend aus der natürlichen Sprache und üblichen mathematischen Ausdrücken ableiten. [BG2008]

- Im Bereich der mikrocontrollerbasierten Systeme ist C die am weitesten verbreitete Sprache, was unter anderem durch einen guten Kompromiss zwischen Lesbarkeit und generiertem Overhead begründet ist. [BST2010]
- Trotz der Existenz vieler Dialekte orientieren sich diese an standardisiertem C. Die grundlegende Syntax ist daher plattformunabhängig.

Grundsätzlich können natürlich auch andere Sprachen genutzt werden. Um jedoch in das vorliegende hybride Konzept aus graphischer und textueller Programmierung zu passen, sollte die genutzte Sprache möglichst einfach lesbar sein.

Bestandteile des hybriden Ansatzes

Um einen solchen Ansatz realisieren zu können, müssen verschiedene Bestandteile existieren:

- Eine Bibliothek von Befehlen in graphischer Form,
- ein Quelltexteditor und
- Werkzeuge, um die Arbeit mit den benötigten Variablen zu erleichtern.

Im vorgestellten hybriden Ansatz kann die datenflussorientierte Funktionsbausteinbibliothek aus der zweiten Ebene zur Modellierung des neuen benutzerdefinierten Funktionsbausteins herangezogen werden. Dies hat den Vorteil, dass die Funktionsbausteine dem Nutzer bereits bekannt sind, so dass er sich nicht in eine neue Funktionsbausteinbibliothek einarbeiten muss. Hinzu kommt, dass auf diese Weise auch in einer vorherigen Sitzung erstellte nutzerdefinierte Funktionsbausteine weitergenutzt werden können.

Die Funktionsbausteine der zweiten Ebene sind auf die Modellierung datenflussorientierter Programme ausgelegt. Hier soll jedoch ablauforientiert modelliert werden. Um dies zu ermöglichen, muss die Funktionsbausteinbibliothek um die Kontrollstrukturen der Wiederholung (z.B. for-Schleife) und der bedingten Ausführung (z.B. if-Anweisung) erweitert werden.

Eine mögliche Implementierung mit einer geeigneten Anordnung der Bestandteile ist in Abbildung 54 dargestellt.

Funktionsweise

Ziel ist es, im Quelltexteditor den Quelltext für den neu zu definierenden Funktionsbaustein zu erstellen. Hierfür werden die Quelltexte der bereits existierenden Funktionsbausteine in der gewünschten Reihenfolge im Editor angeordnet, so dass diese später schrittweise abgearbeitet werden können.

Um den Quellcode eines Funktionsbausteins auf einfache Weise einzufügen, steht die Funktionsbausteinbibliothek zur Verfügung. Hier muss lediglich der gewünschte Baustein gesucht und ausgewählt werden. Auf diese Weise wird der zugehörige Quellcode mit nur wenigen Aktionen an die vom Nutzer vorgegebene Position im Quelltexteditor eingefügt.

Genau wie dies bei der graphischen Modellierung der Fall ist, besteht teilweise der Bedarf, die Datentypen der genutzten Variablen des eingefügten Funktionsbausteins festzulegen. Hier bietet sich die Nutzung eines zusätzlichen Dialogs an, welcher den Nutzer bei der Auswahl des korrekten Datentyps unterstützt. Dabei muss berücksichtigt werden, dass eine Variable unter Umständen auch mehrere Datentypen unterstützen kann.

Um dem neuen nutzerdefinierten Funktionsbaustein eine sinnvolle Funktion zuzuweisen, reicht es allerdings nicht aus, die Datentypen der Variablen festzulegen. Weiterhin müssen Verknüpfungen zwischen den einzelnen Variablen der eingefügten Quellcodeabschnitte hergestellt werden. In der graphischen Darstellung der zweiten Ebene der Programmierung wird dies durch die Verbindung eines Ausgangsverbinders eines Funktionsbausteins mit einem Eingangsverbinder eines folgenden Funktionsbausteins realisiert. In der textuellen Darstellung der dritten Ebene muss dieses Problem auf andere Weise gelöst werden. An dieser Stelle werden zwei unterschiedliche Ansätze vorgestellt, welche im Verlauf der Entwicklung analytisch mit Hilfe von Walkthrough-Verfahren (siehe Abschnitt 2.3.5) verglichen wurden.

Der Vergleich mit Hilfe des kognitiven Walkthroughs erfolgte in Form eines explorativen Vorgehens, wobei der wahrscheinliche Handlungsablauf des Nutzers mit dem vom Entwickler vorgegebenen Handlungsablauf abgeglichen wurde und mögliche Fehlerquellen gesucht wurden [LW1997; Pat2000].

Der erste Ansatz beinhaltet eine zusätzliche graphische Darstellung, in welcher der Nutzer die Variablen miteinander verbinden kann, so wie er dies bereits aus dem Datenflussplan kennt. Der zweite Ansatz sieht vor, dass sich der Nutzer selbst vollständig um die Verwaltung der Variablen kümmern muss, wobei ihm hierfür einige Werkzeuge zur Verfügung gestellt werden.

Der erste und rein graphische Ansatz wurde schnell als ungeeignet eingestuft, da die Kombination aus Kontrollstruktur- und Datenflussfunktionsbausteinen nur durch eine komplizierte mehrstufige Darstellung visualisiert werden könnte, was wiederum die Fehleranfälligkeit der Nutzung erhöhte. Außerdem erwies sich diese Darstellung nur als konsistent, solange keine manuellen Änderungen im Quelltext vorgenommen wurden. Ohne die Möglichkeit solcher Änderungen reduziert sich jedoch die Flexibilität bei der Programmierung.

Der zweite Ansatz ist dann gut nutzbar, wenn dem Nutzer verschiedene Werkzeuge zur Verfügung gestellt werden. Hierzu gehören Werkzeuge:

- zur Zuordnung vorhandener Variablen zu den Ein- und Ausgängen des zu erstellenden Funktionsbausteins,
- zum Umbenennen von Variablen,
- zum Ersetzen einer Variable durch eine andere Variable und
- zur Analyse des Quellcodes.

Werden diese Bearbeitungswerkzeuge konsequent genutzt, können Verknüpfungsfehler zwischen den Variablen vermieden werden, wobei dennoch eine hohe Flexibilität in der Programmierung gewährleistet ist. Es bleibt jedoch festzuhalten, dass durch die textuelle Form immer noch ein erhöhtes Fehlerpotential vorliegt, weshalb komplexere Berechnungen nur von erfahrenen Nutzern realisiert werden sollten, da die Entwicklung eines fehlerhaften Funktionsbausteins zu Fehlern im Datenflussplans führt. Ein Werkzeug zur Quellcodeanalyse kann die Wahrscheinlichkeit von Fehlern zwar reduzieren, jedoch kann auch dieses nicht alle Fehler finden. So werden logische Fehler, wie zum Beispiel das Vertauschen eines Plus mit einem Minus, gewöhnlich nicht erkannt.

Sonstige Maßnahmen

Dem Nutzer steht mit dem oben vorgestellten Konzept bereits ein funktionsfähiges Entwicklungssystem zur Verfügung. Dennoch sind hier verschiedene Erweiterungen denkbar, um die Flexibilität zu erhöhen und die Nutzerunterstützung zu verbessern. So ist es beispielsweise sinnvoll, für erfahrene Nutzer die Möglichkeit des Einbindens zusätzlicher Bibliotheken zuzulassen. Aber auch eine assistentenbasierte Unterstützung beim Implementieren serieller Datenverbindungen, um beispielsweise kostengünstige Sensoren abfragen zu können, ist vorstellbar.

Kritische Betrachtung des vorgestellten Konzeptes

Zusammengefasst werden in einer solchen Oberfläche die folgenden Elemente kombiniert:

- graphische Datenfluss- und Kontrollstrukturfunktionsbausteine,
- textueller Quellcode,
- Variablenmanipulation durch verschiedene Werkzeuge,
- Werkzeuge zur Fehlersuche im Quellcode und
- weitere Werkzeuge zur Integration zusätzlicher Bibliotheken und anderer komplexer Funktionen

Daraus resultieren verschiedene Vorteile:

- Durch die Möglichkeit Funktionsbausteine für die Modellierung im Datenflussplan zu erzeugen, erhöht sich die Flexibilität der Programmierung für den Anwender. Vor allem können domänenspezifische Funktionsbausteine erstellt werden, welche in späteren Aufgaben regelmäßig wiederverwendet werden können und somit die Entwicklung deutlich beschleunigen können.

- Für die Erstellung des nutzerdefinierten Funktionsbausteins sind keine oder nur wenige Programmierkenntnisse nötig. Der Nachteil einer jeden textuellen Programmiersprache ist, dass der Nutzer zunächst die Syntax erlernen muss, um überhaupt Quellcode schreiben zu können. Beim vorgestellten Konzept entfällt dies. Der Nutzer muss den Quellcode und seine grundlegende Syntax lediglich verstehen können, wenn er ihn liest. Dies kann durch das Einfügen von Kommentaren weiter vereinfacht werden.
- Der Nutzer kann, während er den Funktionsbaustein erstellt, grundlegendes Wissen über die textuelle Programmierung erlangen. Dies beinhaltet sowohl grundlegende Programmierstrukturen als auch die Syntax der genutzten Sprache. Natürlich werden auf diese Weise keine umfassenden Programmierkenntnisse vermittelt. Dennoch kann das so erlangte Wissen den Einstieg in die textuelle Programmierung erleichtern.
- Durch die Möglichkeit der Integration vorhandener Bibliotheken können erfahrene Nutzer bereits entwickelte Software weiternutzen und diese anderen Nutzern in Form von Funktionsbausteinen zur Verfügung stellen.

Nachteile des Konzepts:

- Komplexe Funktionsbausteine, welche beispielsweise zusätzliche Hardware auslesen sollen, können auch bei diesem Ansatz nur von Entwicklern mit Expertenwissen implementiert werden. Allerdings können diese Nutzergruppen vergleichsweise einfach neue Funktionsbausteine entwerfen und anderen Nutzern zur Verfügung stellen.
- Da die nutzerdefinierten Funktionsbausteine ablauforientiert programmiert werden, können diese einen negativen Einfluss auf die Echtzeitfähigkeit des erstellten Mikrocontrollerprogramms haben. Ist Echtzeitfähigkeit vorausgesetzt, muss der Nutzer selbst darauf achten, dass diese Randbedingung durch seinen entwickelten Funktionsbaustein eingehalten wird.

4.2 Modellierung ereignisgesteuerter Abläufe

Gerade im Hobbybereich ist es wichtig, dass ereignisgesteuerte Abläufe modelliert werden können. Dies entspricht sehr gut der weit verbreiteten Vorstellung von Aktion und Reaktion und ist daher sehr leicht nachvollziehbar. Dieses Prinzip lässt sich sehr gut mit Hilfe eines Ablaufdiagramms modellieren [DIN61131-3-2003]. Deshalb werden Varianten des Ablaufdiagramms, wie bereits in Abschnitt 4.1 beschrieben, in vielen Entwicklungsumgebungen auf der obersten Entwicklungsebene eingesetzt. Auch in EasyLab ist dies der Fall. Da die Nutzer in EasyLab jedoch teilweise Probleme hatten, die ereignisgesteuerten Vorgänge mit Hilfe des Ablaufdiagramms zu realisieren, wurden Untersuchungen vorgenommen, wie die Modellierung vereinfacht werden kann, ohne dabei jedoch die Flexibilität einzuschränken.

Erkannte Probleme

Die in Abschnitt 3.3.3 bereits vorgestellten Probleme bei der Entwicklung eines Mikrocontrollerprogramms mit ereignisgesteuerten Abläufen lassen sich nach ihrer Signifikanz wie folgt einteilen:

- Die Nutzer wussten nicht, dass das Ablaufdiagramm überhaupt existierte.
- Die Nutzer erkannten nicht, dass die Möglichkeit bestand, die Ausführungsdauer eines Zustands zu verändern. Dies ist jedoch ein wichtiger Punkt, um Abläufe mit festen zeitlichen Vorgaben, wie zum Beispiel Tastzeiten, modellieren zu können.
- Das Einfügen neuer Zustände und Verzweigungen in das Ablaufdiagramm wurde von einigen Nutzern als wenig intuitiv angesehen.

- Während des Ausführens eines in EasyLab entwickelten Mikrocontrollerprogramms wird der gerade aktive Zustand des Ablaufdiagramms so lange wiederholt, bis eine Bedingung an einem der angeschlossenen Ausgangsverbinder erfüllt ist. Die Nutzer verstanden zwar die Funktion der Bedingungen, nicht aber, dass der aktive Zustand immer wieder wiederholt wird.

Der letzte Punkt ist als eher unkritisch anzusehen, da spätestens während der Simulation oder des Onlinedebuggings der Ablauf des Programms vom Nutzer erkannt wird.

Der dritte Punkt repräsentiert ein typisches Problem der Nutzerfreundlichkeit, weshalb eine Lösung des Problems anzustreben ist.

Wirklich kritisch sind der erste und der zweite Punkt. Erkennt der Nutzer nicht, dass das Ablaufdiagramm existiert oder dass die Ausführungsdauer angepasst werden kann, so können ereignisgesteuerte Abläufe nur schwer realisiert werden. Da diese aber, wie bereits beschrieben, im Hobbybereich oft auftreten, muss eine Lösung gefunden werden.

Die in EasyLab implementierte Darstellungsform der Ablaufsprache ist auf eine Programmierung auf Mikrocontrollerebene ausgelegt. Es können für jeden einzelnen Zustand innerhalb des Ablaufdiagramms Bedingungen zum Verlassen des Zustandes und eine Mindestausführungsdauer festgelegt werden. In anderen Varianten des Ablaufdiagramms, wie zum Beispiel in LEGO MINDSTORMS NXT, können vergleichbare Randbedingungen nur mit zusätzlichem Aufwand implementiert werden. Bei der Programmierung auf Mikrocontrollerebene sind diese Abläufe jedoch oft wichtig, um ereignisgesteuerte Vorgänge zu realisieren. Es ist daher sinnvoll, die Sprache grundsätzlich beizubehalten. Jedoch sollte die Darstellung und die Erstellung des Ablaufdiagramms angepasst werden, um die oben genannten Probleme zu lösen. Die im Folgenden vorgestellten Lösungen wurden in der Diskussion mit Testnutzern erarbeitet, nachdem die in Abschnitt 3.3 vorgestellten Tests durchgeführt wurden.

Lösungsansätze

Zunächst ist es wichtig, dass die Nutzer die Existenz des Ablaufdiagramms erkennen. In EasyLab wird dem Nutzer ein bereits funktionsfähiges Ablaufdiagramm vorgegeben. Da sich dieses in einem Reiter im Hintergrund befindet und für die Erstellung eines Mikrocontrollerprogramms, welches keine ereignisgesteuerten Abläufe benötigt, nicht manipuliert werden muss, ist seine Existenz nicht sofort ersichtlich. Zur Lösung des Problems sollte das Ablaufdiagramm während der Programmierung immer im Vordergrund erkennbar sein. Zudem sollte kein bereits funktionsfähiges Ablaufdiagramm vorgegeben werden. Besser ist es, dieses vom Nutzer selbst erstellen zu lassen, wobei ihm eine dialogbasierte Unterstützung zur Seite gestellt wird. Auf diese Weise kann auch ein unerfahrener Nutzer sehr schnell die Vorteile und Einsatzmöglichkeiten der ereignisgesteuerten Programmierung verstehen.

Ein zusätzlicher Vorteil dieses Ansatzes ist, dass im Rahmen der dialogbasierten Unterstützung auch die Ausführungszeit abgefragt werden kann. Somit ist auch dieses Problem bereits gelöst. Allerdings sind hierfür auch andere Ansätze denkbar, wie zum Beispiel die Anzeige der Ausführungsdauer direkt im Ablaufdiagramm.

Um das Erstellen des Ablaufdiagramms zu erleichtern, ist es zweckmäßig, die Darstellung der einzelnen Elemente zu überdenken. In EasyLab werden die Zustände und die zugehörigen Verbinder, wie in der Informatik üblich, als separate Bestandteile betrachtet. In Abbildung 34 ist ein Beispiel-Ablaufdiagramm aus EasyLab mit den zugehörigen Bearbeitungswerkzeugen (links neben dem Diagramm) gezeigt. Um das Ablaufdiagramm zu verändern, muss der Nutzer zunächst die Stelle auswählen, an welcher er ein neues Element, also einen Zustand oder einen Verbinder, einfügen möchte. Anschließend wählt er das entsprechende Werkzeug aus, wodurch das gewünschte Element eingefügt wird. Die umliegende Struktur passt sich automatisch an.

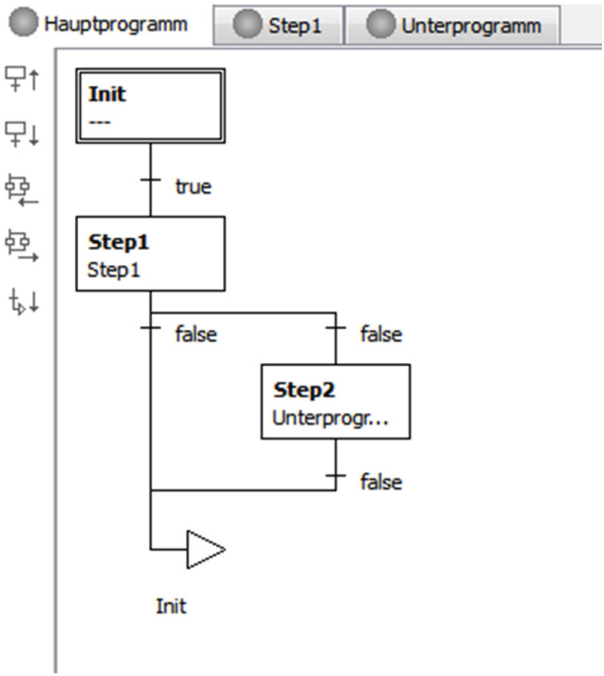

Abbildung 34: Ablaufdiagramm in EasyKit

Dieses Vorgehen bei der Erstellung des Ablaufdiagramms ist aus einer informatiknahen Sichtweise abgeleitet. Sie bietet den Vorteil, dass aus dem so erzeugten Diagramm sehr leicht effizienter Quellcode erzeugt werden kann. Die Diskussion mit den Testnutzern ergab jedoch, dass eine freie Gestaltung des Ablaufdiagramms, zum Beispiel mit Drag-and-Drop-Methoden, vorzuziehen sei. Um dies möglichst einfach zu realisieren, wird die separate Betrachtungsweise der Zustände und Verbinder aufgegeben und jeweils ein Zustand und die zugehörigen anschließenden Verbinder werden als eine Einheit betrachtet. Dies ermöglicht eine einfache dialogbasierte Unterstützung.

Eine solche dialogbasierte Unterstützung biete verschiedene Vorteile, da mit ihrer Hilfe auch Sachverhalte, welche für Einsteiger nur schwer verständlich sind, leichter dargestellt werden können. So kann der Nutzer beispielsweise

bei der Eingabe der Bedingungen an den Ausgangsverbindern der Zustände unterstützt werden, wodurch fehlerhafte Eingaben vermieden werden können. Außerdem kann der Nutzer beim Einfügen eines neuen Zustands in das Ablaufdiagramm dialogbasiert auswählen, ob er einen bereits vorhandenen Datenflussplan mit dem Zustand verknüpfen möchte oder ob automatisch ein neuer Datenflussplan angelegt und dem Zustand zugeordnet werden soll. Ein solches Vorgehen veranschaulicht dem Nutzer den Zusammenhang zwischen dem Ablaufdiagramm und den Datenflussplänen. Eine mögliche Implementierungsvariante des Ablaufdiagramms ist in Abbildung 46 dargestellt.

Quellcodeerzeugung

Wie bereits beschrieben, stellt das Ablaufdiagramm die obere Ebene in der Programmierung dar. Seine Beschaffenheit ist daher verantwortlich für den grundsätzlichen strukturellen Aufbau des zu entwickelnden Mikrocontrollerprogramms. Die Datenflusspläne bestimmen hingegen die Funktionalität einzelner Elemente des Programms. Dies entspricht dem typischen Aufbau einer Software, welche nach dem Paradigma der strukturierten Programmierung erstellt wurde. Mit der Programmiersprache C, welche hier als Mittel der Wahl angesehen wird, lässt sich eine strukturierte Programmierung gut realisieren. Wird mit Hilfe der Quellcodeerzeugung aus der graphischen Darstellung ein textuelles Programm erstellt, so ist es daher sinnvoll die Trennung zwischen dem Ablaufdiagramm und den Datenflussplänen beizubehalten.

Aus der Struktur des Ablaufdiagramms wird daher die „main“-Funktion des C-Programms generiert. Diese Funktion stellt den Einsprungspunkt dar und wird somit beim Start des Programms ausgeführt. Am Anfang der „main“-Funktion müssen die folgenden Aktionen ausgeführt werden:

- Deklaration der globalen Variablen
- Zuweisen der Standardwerte der globalen Variablen
- Initialisierung des Timers
- Initialisierung weiterer Komponenten, wie der Interruptroutinen und der Kommunikation

Anschließend wird eine Schleife eingefügt, deren Abbruchbedingung nie erfüllt wird, so dass sie nicht wieder verlassen wird. In dieser Schleife wird die eigentliche Struktur des Ablaufdiagramms unter Berücksichtigung der Weiterleitungsbedingungen und der zeitlichen Restriktionen der Zustände erzeugt. Als erstes wird dabei immer der Initialisierungszustand ausgeführt. Nach jedem Ausführen eines Zustands muss zunächst geprüft werden, ob die vom Nutzer voreingestellte Ausführungszeit bereits erreicht wurde. Ist dies nicht der Fall, so wartet der Mikrocontroller an dieser Stelle. Liegt die vom Nutzer gewünschte Ausführungszeit oberhalb der tatsächlichen Ausführungszeit des Zustands, so kann auf diese Weise eine bestimmte Tastzeit vorgegeben werden, was beispielsweise in der Regelungs- und Automatisierungstechnik häufig benötigt wird. Weiterhin wird getestet, ob eine Bedingung an einem Ausgang des Zustands erfüllt ist. Ist dies der Fall, so wird der an diesem Ausgang angeschlossene Zustand für die Ausführung im nächsten Durchlauf vorgemerkt. Ist dies nicht der Fall, so wird der aktuelle Zustand erneut ausgeführt.

Die Ausführung eines Zustands bedeutet gleichzeitig die Ausführung des ihm zugeordneten Datenflussplans. Für jeden Datenflussplan muss dabei eine neue Funktion angelegt werden, welche aus der „main“-Funktion heraus ausgeführt wird. Der Quellcode kann hier vergleichsweise einfach und effizient generiert werden, indem im Datenflussplan untersucht wird, welche Funktionsbausteine entweder in die Berechnung von globalen Variablen oder von hardwarenahen Funktionsbausteinen eingehen. So entstehen Berechnungsketten, deren Quellcode anschließend nur noch in die Funktion eingefügt werden muss. Dabei müssen Überschneidungen zwischen unterschiedlichen Berechnungsketten berücksichtigt und die Variablen der Ausgänge der Funktionsbausteine mit den Variablen des Eingangs des jeweils folgenden Funktionsbausteins gleichgesetzt werden.

4.3 Integrierte Entwicklung von Hard- und Software

Während bei der Entwicklung moderner PC-Software oft keine Betrachtung der angeschlossenen Peripherie des Computers notwendig ist, muss dieser Aspekt bei der Entwicklung mikrocontrollerbasierter mechatronischer Systeme berücksichtigt werden. Diese Herangehensweise wird heute als integrierte Hardware/Software-Entwicklung bezeichnet. Die meisten Hersteller von Entwicklungsplattformen für mechatronische Systeme beachten dies, indem sie ihre Plattform modular aufbauen, so dass der Mikrocontroller mit seinen Schaltkreisen ein zentrales Modul darstellt, welches mit vorgefertigten Peripheriemodulen erweitert werden kann. Die Peripheriemodule beinhalten verschiedene Sensoren und Aktoren, so dass ein mechatronisches System aufgebaut werden kann. Beispiele hierfür wurden in Abschnitt 2.4 vorgestellt. Ein solches Vorgehen widerspricht jedoch der hier gewünschten Zielstellung der Plattformunabhängigkeit. Daher muss im vorliegenden Fall bei der integrierten Hardware/Software-Entwicklung von rein modulbasierten Ansätzen Abstand genommen werden. Nur so können unterschiedliche Plattformen mit einer Vielzahl unterschiedlicher Sensoren und Aktoren kombiniert werden. Dies bedeutet jedoch, dass der Nutzer bei der Schaltungsentwicklung unterstützt werden muss, um den Mikrocontroller zum einen überhaupt in Betrieb nehmen zu können und zum zweiten die Zerstörung von Komponenten des Systems durch unsachgemäße Verschaltung zu verhindern. Vor allem das Herstellen elektrischer Verbindungen zwischen Sensoren, Aktoren und dem Mikrocontroller ist sehr fehleranfällig und kann bei falschem Aufbau zur Zerstörung angeschlossener Bauteile führen. Um auch Nutzern ohne elektrotechnisches Fachwissen die Nutzung des Systems zu ermöglichen, ist es notwendig, vor solchen Fehlern zu warnen und Vorschläge für funktionsfähige Schaltungen zu liefern.

4.3.1 Modellierung der Entwicklungsplattform

Zentraler Bestandteil eines modernen mechatronischen Systems ist die informationstechnische Plattform. Darunter versteht man ein elektronisches System, in welchem eine Datenverarbeitung stattfindet, also beispielsweise

ein Mikrocontroller, ein Evaluierungsboard, ein Industrie-PC oder eine Speicherprogrammierbare Steuerung. Ihre Leistungsfähigkeit und ihre Peripherie haben einen entscheidenden Einfluss auf den möglichen Aufbau des mechatronischen Systems. Daher muss der Nutzer die unterschiedlichen Hardwareplattformen beim Anlegen des Projekts auswählen können. Die Auswahl kann dabei eingeschränkt werden, indem der Nutzer vorgibt, wie viele analoge und digitale Signale im System verarbeitet werden sollen, woraufhin nichtkompatible Plattformen ausgeblendet werden. Um eine möglichst hohe Flexibilität zu gewährleisten, müssen dabei auch verschiedene Plattformtypen unterstützt werden. Das bedeutet, dass neben einfachen Mikrocontrollern auch Evaluierungs- und Testboards berücksichtigt werden müssen.

Im industriellen Bereich ist oft die Flexibilität vorhanden, um zwischen unterschiedlichen Plattformen mit unterschiedlicher Peripherie wechseln zu können. Diese Möglichkeit besteht im Hobbybereich oft nicht. Beim Einstieg in die Programmierung von Mikrocontrollern erfolgt hier die Auswahl des ersten Mikrocontrollers oft durch Empfehlungen Dritter. Meist nutzen die Anwender Mikrocontroller dieser Familie später weiter, da sie beispielsweise nur einen bestimmten Programmieradapter zur Verfügung haben. Aus diesem Grund muss dem Nutzer beim Anlegen eines neuen Projekts eine möglichst große Auswahl unterschiedlicher Plattformen verschiedener Hersteller zur Verfügung gestellt werden.

Das in der Software hinterlegte Modell der Plattform muss verschiedene Angaben über dessen nutzbare Funktionalität beinhalten, da sie Auswirkungen auf die Kompatibilität zu Sensoren und Aktoren hat. Folgende Informationen müssen vorhanden sein:

- Anzahl der nutzbaren Anschlüsse der Plattform
- Unterstützte Spannungen und Spannungsbereiche
- Maximale bereitstellbare Stromstärke

Und für jeden einzelnen Anschluss der Plattform:

- Mögliche Richtung (Input, Output, Beide)

- Integrierte Peripherie (Analog-Digital-Wandler, Schaltung zur Erzeugung eines pulsweitenmodulierten Signals, Flankenzähler, …)
- Existenz von Pull-Up- oder Pull-Down-Widerständen

Für die integrierte Peripherie müssen weitere Informationen über die Weiterverarbeitung der Signale bereitgestellt werden. Beim Analog-Digital-Wandler ist dies beispielsweise die Berechnungsvorschrift, wie aus dem Spannungssignal ein digitalisierter Wert entsteht, welcher in Form eines unterstützten Datentyps innerhalb der Entwicklungsumgebung weitergenutzt werden kann.

4.3.2 Inbetriebnahme und Programmierung der informationstechnischen Plattform

In dieser Arbeit werden Mikrocontroller und mikrocontrollerbasierte Evaluierungsboards als informationstechnische Plattform genutzt. Diese besitzen meist die Eigenschaft, dass sie nur mit Hilfe zusätzlicher Schaltungen programmiert und in Betrieb genommen werden können. Daher muss dem Nutzer das Wissen zur Erstellung dieser Schaltungen zur Verfügung gestellt werden.

Die Informationen müssen für Nutzer ohne Expertenwissen aufbereitet werden, wobei die folgenden Inhalte wichtig sind:

- Schaltpläne für die Programmierung und den Betrieb des Mikrocontrollers
- Legende für alle im Schaltplan genutzten Komponenten (Kondensatoren, Widerstände, Quarze, …) mit Schaltsymbol, Bezeichnung und eventuell Bild
- Informationen über die grundlegenden Funktionen der genutzten Komponenten
- Beschriftung der Pins des Mikrocontrollers, so dass der Nutzer die Verbindung zwischen der Hardware und der Repräsentation der Hardware in der Entwicklungsumgebung erkennt
- Informationen über nutzbare Programmieradapter

- Informationen über Anforderungen an die Spannungsversorgung
- Allgemeine Informationen über die Handhabung elektronischer Bauteile, zum Beispiel in Bezug auf statische Aufladung

Die Informationen werden in diesem Umfang nur benötigt, wenn ein reiner Mikrocontroller als Plattform genutzt wird. Wird hingegen ein Evaluierungsboard eingesetzt, so beinhaltet dieses im Normalfall bereits die wichtigsten Schaltungen. Die zur Inbetriebnahme benötigten Informationen reduzieren sich daher auf ein Minimum.

4.3.3 Modellierung der Sensoren und Aktoren

Um den Nutzer beim Anschließen von Sensoren und Aktoren unterstützen zu können, muss die Möglichkeit bestehen, Hardware innerhalb der Entwicklungsumgebung nachzubilden. Das bedeutet, dass einzelne Hardwarebestandteile in Form einer Bibliothek zur Verfügung gestellt werden, so dass der Nutzer diese frei zusammenstellen und miteinander verbinden kann, um so die Funktionalität der Hardware zu modellieren. Wirklich relevant sind dabei allerdings nicht die mechanischen, optischen oder chemischen Eigenschaften der Hardware, sondern nur die elektrischen Schnittstellen der Sensoren und Aktoren.

4.3.3.1 Testaufbauten aus dem Hobby- und Freizeitbereich

Um einen Überblick zu erhalten, welche Schnittstellen im Hobby- und Freizeitbereich oft genutzt werden, wurde mit Unterstützung von Nutzern aus diesem Bereich eine Vielzahl von Beispielanwendungen zusammengetragen und untersucht. Davon wurden einige Versuche zu Testzwecken realisiert. Diese werden im Folgenden kurz vorgestellt.

Gewächshausdemonstrator

In Gewächshäusern lassen sich grundsätzlich sehr viele Anwendungsmöglichkeiten für die Regelungstechnik finden. Der Einsatz von mikrocontrollerbasierten Systemen erscheint hier sinnvoll, da oft nur geringe Datenmengen verarbeitet werden müssen. Im industriellen Bereich existiert bereits eine Vielzahl unterschiedlicher Lösungen. Im Hobbybereich ist dies nur teilweise der Fall. So stehen verschiedene Bewässerungssysteme zu bezahlbaren Preisen zur Verfügung. Verdunklungssysteme zum Schutz gegen

übermäßige Sonneneinstrahlung sind jedoch noch vergleichsweise teuer, obwohl sie aus nur wenigen handelsüblichen Komponenten bestehen. Im Demonstrator eines Gewächshauses, welches in Abbildung 35 dargestellt ist, wurde daher ein solches vollautomatisiertes Verdunkelungssystem installiert.

Es wird ein kostengünstiger Lichtsensor genutzt, welcher oft im Hobbybereich eingesetzt wird. Die Verdunklung erfolgt durch eine handelsübliche Jalousie. Diese kann zum einen durch einen Motor geöffnet und geschlossen werden, zum anderen können die Lamellenwinkel durch einen weiteren Motor angepasst werden. Als informationstechnische Plattform wird die industrielle Version von EasyKit genutzt.

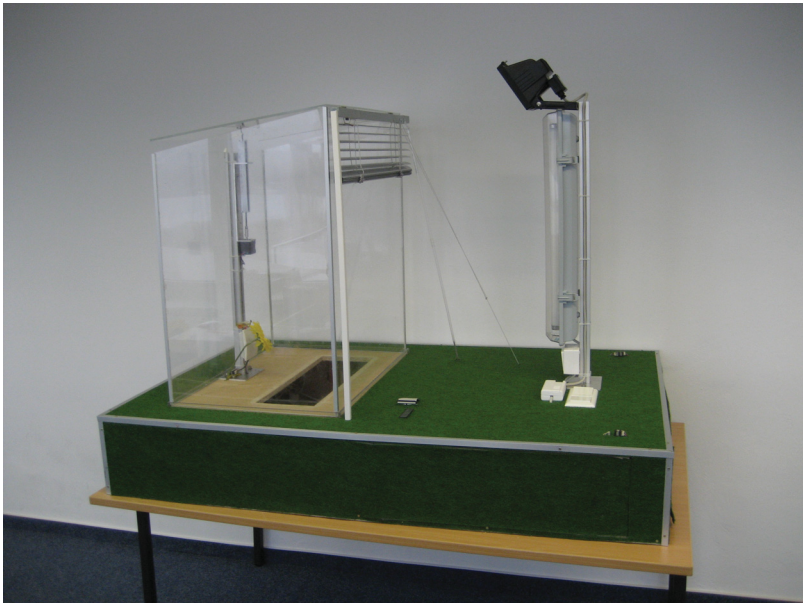

Abbildung 35: Gewächshausdemonstrator zur Regelung des Lichteinfalls

Raumklimaüberwachungssystem

In Gesprächen mit Nutzern aus dem Hobby- und Freizeitbereich zeigte sich, dass diese bereits oft den Gedanken hatten, das Raumklima innerhalb ihrer Wohnungen und Häuser zu regeln, wobei vor allem der Wunsch nach einer ausgewogenen relativen Luftfeuchte, einer angenehmen Temperatur und einer angemessenen Frischluftzufuhr bestand. Allerdings hatte aufgrund der nicht vorhandenen oder zu teuren Aktorik noch keiner der Befragten ein solches Regelungssystem realisiert. Die Überwachung der Raumklimawerte war hingegen weit verbreitet. Dabei konzentrierten sich die Nutzer aus dem Hobby- und Freizeitbereich vorrangig auf die Messung der Luftfeuchte und der Temperatur, da hierfür mit dem SHT11 der Firma Sensirion [Sen2013] ein sehr kostengünstiger Sensor zur Verfügung steht, welcher durch verschiedene frei im Internet erhältliche Softwarebibliotheken vergleichsweise einfach in Betrieb zu nehmen ist.

Dieser Sensor wurde von fast allen befragten Nutzern eingesetzt. Der Bedarf an Frischluft wurde jedoch nicht überwacht, was wiederum durch die vergleichsweise hohen Preise der benötigten Sensoren begründet wurde. Es wurden daher zwei unterschiedliche Systeme zur Raumklimaüberwachung aufgebaut. Das erste besteht aus dem genannten Sensor von Sensirion, während das zweite einen kombinierten Kohlenstoffdioxid-, Luftfeuchte- und Temperatursensor besitzt. Das zweite System ist in Abbildung 36 dargestellt.

Der Sensor des dargestellten Systems stellt für jede der drei physikalischen Größen eine analoge 0…10 V Schnittstelle bereit. Diese wird vom Mikrocontroller ausgelesen und über eine serielle Schnittstelle an das Display weitergeleitet. Das Display entstand in der vorliegenden Version noch im Eigenbau. Die Erfahrungen mit Nutzern im industriellen und im Hobbybereich zeigten, dass solche oder vergleichbare Darstellungsmöglichkeiten für Daten in vielen Anwendungen benötigt werden. Daher wurde für EasyKit ein Modul zum Anschluss einer Flüssigkristallanzeige entwickelt.

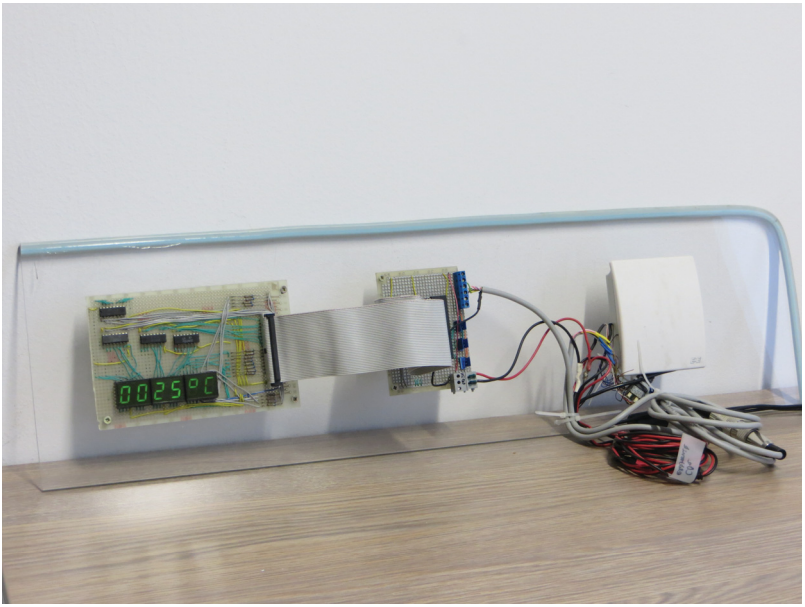

Abbildung 36: Raumklimaüberwachungssystem zur Messung und Anzeige von Kohlenstoffdioxidgehalt, relativer Luftfeuchte und Temperatur

Modelleisenbahn

Eine der befragten Personen hatte in seinem Keller eine Modelleisenbahnlandschaft errichtet. Diese bestand aus drei elektrisch voneinander unabhängigen Gleisabschnitten, welche sich untereinander kreuzten. Über Lichtschranken und Transistorschaltungen wurden die Versorgungsspannungen der einzelnen Stromkreise so an und ausgeschaltet, dass Kollisionen vermieden werden konnten.

Diese komplexe Logikschaltung wurde im Rahmen der hier vorgestellten Arbeit durch einen Mikrocontroller ersetzt. Dies ermöglichte eine deutliche Erweiterung des realisierbaren Funktionsumfangs der Elektronik, so dass zum Beispiel die Geschwindigkeit der Züge über Pulsweitenmodulation gesteuert werden konnte.

Heimdiskobeleuchtungssystem

Für eine private Zimmerdisko wurde ein Beleuchtungssystem mit einer Diskokugel gebaut. Die im Spielwarenhandel gekaufte Diskokugel wurde an einen Gleichstrommotor montiert. Der Motor und mehrere lichtemittierende Dioden (LEDs) unterschiedlicher Farben sind mit dem Mikrocontroller verbunden. Dieser ist so programmiert, dass die LEDs und der Motor in vorgegebenen Zeitabständen an- und ausgeschaltet werden. Über Schalter kann zwischen unterschiedlichen Beleuchtungssequenzen gewählt werden.

4.3.3.2 Akademische und industrielle Testaufbauten

Neben den oben vorgestellten Anwendungen aus dem Freizeit- und Hobbybereich wurden auch einige akademische und industrielle Anwendungen realisiert und untersucht, welche im Folgenden kurz beschrieben werden.

Pumpentestsystem

In einem Testsystem sollten Pumpen zur Förderung verschiedener Flüssigkeiten unter variierenden realistischen Lastbedingungen geregelt werden. In Abbildung 37 ist der Teststand dargestellt.

Die Pumpe selbst wird über einen Frequenzumrichter angesteuert. Diesem wird ein Signal im Bereich von 0…10 V vorgegeben, nach welcher der Frequenzumrichter am Motor die entsprechende Drehzahl einstellt. Die Lastbedingungen werden durch jeweils ein Proportionalventil auf der Druckseite und der Saugseite realisiert.

Um Tests bei unterschiedlichen Flüssigkeitstemperaturen durchführen zu können, wurde ein Heizelement im System installiert. In sehr kurzer Entfernung zur Pumpe werden auf der Druck- und auf der Saugseite die Drücke mit Hilfe industrieller Drucksensoren gemessen. In etwas größerer Entfernung zur Pumpe befinden sich zwei Temperatursensoren, welche in Verbindung mit dem Heizelement für die Einhaltung der gewünschten Temperatur des Mediums genutzt werden. Außerdem befindet sich auf der Druckseite ein Durchflusssensor. Zum Auslesen der Sensoren und zum Ansteuern der Aktoren wird die elektronische Hardware des industriellen EasyKit-Systems genutzt.

Abbildung 37: Teststand für die Vermessung und Regelung von Pumpen unter variierenden realistischen Lastbedingungen

Dieser Aufbau ermöglicht die Simulation vieler verschiedener Last- und Fehlerfälle, wie zum Beispiel schwankende Abnahmemengen beim Verbraucher oder ein zu langsames Nachfließen des geförderten Mediums auf der Saugseite.

Kalibrier- und Testsystem für Außenluftdurchlässe

In einem Projekt des Fachgebietes Systemanalyse der Technischen Universität Ilmenau wurde ein vollautomatisierter Außenluftdurchlass entwickelt. Dieser ermöglicht die Regelung der Luftwechselrate zwischen Innenräumen und Umgebung. Hierfür wird anhand der im System gemessenen Geschwindigkeit der strömenden Luft der Öffnungswinkel einer Klappe verändert.

Für den Test des Gesamtsystems und für die Kalibrierung des Sensors wurde ein Teststand entwickelt. Während der Testphase wurde der Aufbau genutzt, um Außenluftdurchlässe mit unterschiedlichen Schallschutzinstallationen zu untersuchen. Es wurde getestet, wie sich der Schallschutz auf den Luftwiderstand auswirkt und ob die in der DIN 1946-6 [DIN1946-6-2009] festgesetzten Mindestanforderungen des Luftdurchsatzes bei vorgegebenem Differenzdruck erfüllt sind.

Das Testsystem ist in Abbildung 38 dargestellt. Im unteren Bereich befindet sich die Testkammer mit einem Einschubschlitz für den Außenluftdurchlass. An der Testkammer ist ein Sensor befestigt, welcher den Differenzdruck zwischen Innenraum und Umgebung misst. Zum Absaugen oder Einblasen der Luft wurde ein handelsüblicher PC-Ventilator installiert. Über ein etwa zwei Meter langes Rohr konstanten Durchmessers ist dieser mit der Testkammer verbunden. Auf diese Weise besteht innerhalb des mittleren Bereichs des Rohres eine weitgehend laminare Strömung. Hier kann die Flussgeschwindigkeit sehr präzise bestimmt werden. Fluiddynamische Simulationen zeigten, dass die Strömung in diesem Abschnitt des Rohres, bis auf einige Randzonenbereiche, tatsächlich weitgehend laminar ist. Der Mikrocontroller wertet das Signal des Differenzdrucksensors aus und regelt den Ventilator über ein pulsweitenmoduliertes Signal. Anschließend kann die Strömungsgeschwindigkeit mit einer separaten Sonde gemessen werden.

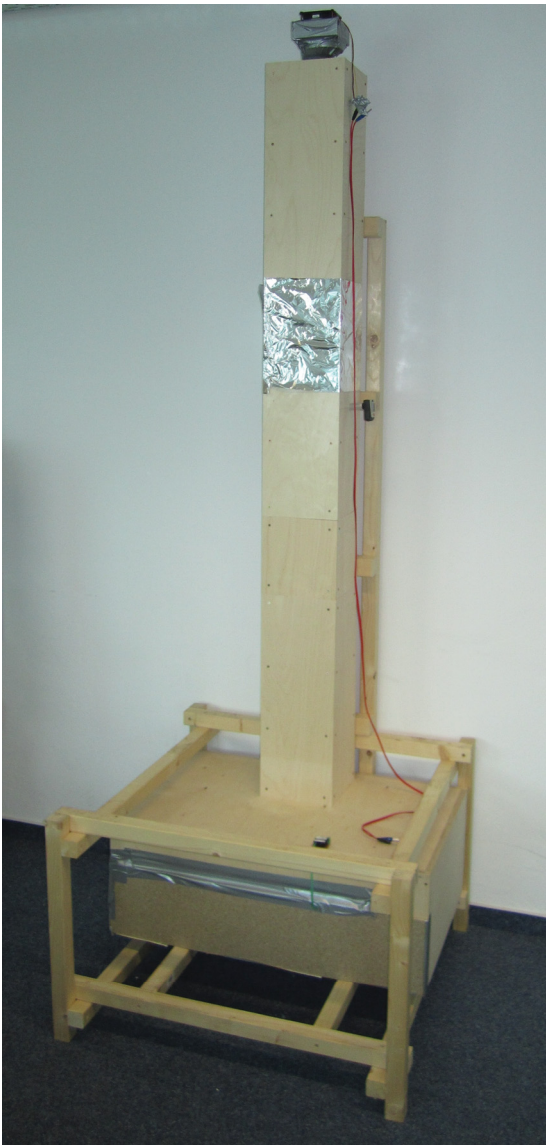

Abbildung 38: Teststand für Außenluftdurchlässe

Bei der Kalibrierung des Strömungssensors erfolgte ein ähnliches Vorgehen. Es wurden unterschiedliche Drehzahlen für den Ventilator und unterschiedliche Stellwinkel für die Klappe eingestellt. Die so erfassten Messkurven beschreiben den Zusammenhang zwischen der gemessenen Spannung des Strömungssensors im Außenluftdurchlass und der gemessenen Luftgeschwindigkeit des Thermoanemometers im laminaren Strömungsbereich. Der Außenluftdurchlass bestimmt mit Hilfe dieser Messkurven die reale Strömungsgeschwindigkeit und somit auch die Luftwechselrate.

Dreitanksystem

In regelungstechnischen Vorlesungen wird das Dreitanksystem immer wieder gern als Beispiel für die Vermittlung von Methoden der Zustandsraumdarstellung gewählt.

Um den Studenten auch die praktische Arbeit hieran zu ermöglichen, wurde ein solches System mit Hilfe des EduKit PA der Firma FESTO Didactic [Fes2013] realisiert. Dieses ist in Abbildung 39 dargestellt.

Ziel beim Dreitanksystem ist es, die Wassermenge in den oberen drei Behältern zu regeln. An diesen wurden hierfür Sensoren zur Messung des Füllstands befestigt. Der untere große Behälter dient lediglich als Reservoir. Aus diesem wird Wasser durch zwei Gleichstrommotoren in die beiden äußeren Behälter gepumpt. Hinter jeder Pumpe befindet sich ein Durchflussmesser, so dass das gepumpte Wasservolumen bestimmt werden kann.

Weiterhin wurden elektrische Schaltventile installiert. Mit diesen können sowohl die Verbindungsrohre zwischen den oberen Behältern als auch der Ablauf in das Wasserreservoir geöffnet und geschlossen werden. Auch die Sensoren und Aktoren dieses Dreitanksystems wurden mit Hilfe von EasyKit in Betrieb genommen und die benötigten elektronischen Erweiterungen wurden dokumentiert.

Abbildung 39: Experimentieranordnung Dreitanksystem für studentische Versuche im Bereich der Regelungstechnik

4.3.3.3 Aktoren und Sensoren der Beispielanwendungen

Die Untersuchung der in Abschnitt 4.3.3.2 vorgestellten realisierten Systeme und die theoretische Betrachtung anderer mechatronischer Systeme ergab eine große Menge möglicher Sensoren und Aktoren, welche mit unterschiedlichen Schnittstellen ausgestattet sind. Diese vollständig integrieren zu

wollen, würde zum einen die Bibliothek unkontrollierbar vergrößern, zum anderen wäre der Aufwand für Implementation und Wartung immens. Daher wurden die Schnittstellenschaltungen zwischen den Sensoren und dem Mikrocontroller beziehungsweise zwischen dem Mikrocontroller und den Aktoren untersucht. Dabei zeigte sich, dass grundsätzlich nur eine geringe Anzahl Schaltungen benötigt wird, welche sich lediglich in einigen wenigen Parametern unterscheiden. In Tabelle 2 und Tabelle 3 sind die Schaltungen mit den zugehörigen Beispielanwendungen aufgeführt.

Ausleseschaltungen für	Gewächshaus-demonstrator	Raumklima-überwachungssystem	Modelleisenbahn	Heimdiskobeleuch-tungssystem	Pumpenteststand	Kalibrier-/Testsystem für Außenluftdurchlässe	Dreitanksystem
Taster/Schalter	x		x	x		x	
Photodioden/-transistoren			x				
Potentiometer	x		x		x		
Sensoren mit serieller Schnittstelle (z.B. SPI, I²C)	x					x	
Sensoren mit standardisierter Schnittstelle: 0…20 mA, 4…20 mA, 0…10 VDC, 2…10 VDC [DIN/IEC60381-1-1985; DIN/IEC60381-2-1980]		x			x	x	x

Tabelle 2: Übersicht über die in den Beispielanwendungen genutzten Ausleseschaltungen (x = wird genutzt)

Ansteuerungs-schaltungen für	Gewächshaus-demonstrator	Raumklima-überwachungssystem	Modelleisenbahn	Heimdiskobeleuch-tungssystem	Pumpenteststand	Kalibrier-/Testsystem für Außenluftdurchlässe	Dreitanksystem
Lichtemittierende Dioden (LEDs)		x	x	x			
Gleichstrommotoren (analog, digital, pulsweitenmoduliert)	x		x	x		x	x
Schrittmotoren	x						
Relais	x		x				
Aktoren mit standardisierter Schnittstelle: 0…20 mA, 4…20 mA, 0…10 VDC, 2…10 VDC [DIN/IEC60381-1-1985; DIN/IEC60381-2-1980]					x		x
Displays		x					

Tabelle 3: Übersicht über die in den Beispielanwendungen genutzten Ansteuerungsschaltungen (x = wird genutzt)

Mit Hilfe dieser Schaltungen und der dazugehörigen Sensoren und Aktoren kann eine Vielzahl unterschiedlicher mechatronischer Systeme aufgebaut werden, wobei die Liste an dieser Stelle noch nicht vollständig sein kann. Es ist jedoch weder möglich noch sinnvoll, Nutzern im Hobbybereich die Anwendung einer jeglichen erdenklichen Sensorik und Aktorik zu ermöglichen. Vielmehr geht es darum, dass der Anwender sich nicht nur auf die Nutzung kostenintensiver industrieller Lösungen beschränken muss. Die oben aufgestellte Liste begnügt sich daher mit einer Auswahl der im Hobbybereich gebräuchlichsten Schaltungen.

Wie bereits beschrieben, ist die Anzahl der aufgeführten Schnittstellenschaltungen relativ gering, da diese jeweils eine ganze Gruppe ähnlicher Schaltungen repräsentieren. Diese unterscheiden sich lediglich in einigen Parametern, welche beim Anschließen der Hardware an den Mikrocontroller berücksichtigt werden müssen. Dies bedeutet im Allgemeinen, dass die Werte für Widerstände oder Kondensatoren an die vorliegenden Spannungs- und Strombedürfnisse angepasst werden müssen. Daher muss der Nutzer neben der Signalart des Sensors oder Aktors weitere Parameter vorgeben. In den meisten Fällen sind dies allerdings nur die benötigte Spannung und Stromstärke. Erst mit diesen Parametern kann ermittelt werden, ob und wie der jeweilige Sensor oder Aktor an den Mikrocontroller angeschlossen werden kann.

Um einordnen zu können, ob die Sensoren und Aktoren mit dem Mikrocontroller kompatibel sind, müssen für diese mindestens die folgenden Informationen bekannt sein:

- Welche Signalart liegt vor? (Strom, Spannung, Bus, Flanken, PWM, analog, digital, …)
- Welche Spannungen oder Spannungsbereiche werden benötigt oder liegen an?
- Welche Stromstärke wird benötigt?
- Ist die Last induktiv?
- Wird grundsätzlich eine Anpassungsschaltung benötigt, auch wenn die Spannungs- und Strombereiche vom Mikrocontroller unterstützt werden?

4.3.4 Zusammenhänge zwischen Hardware und Software

Die analytische Untersuchung der Möglichkeiten der Darstellung der Hardware innerhalb der Entwicklungsumgebung zeigte, dass es verschiedene Abhängigkeiten gibt, welche bei der Realisierung berücksichtigt werden müssen:

- Die angeschlossenen Sensoren und Aktoren bestimmen die zu nutzende Peripherie des Mikrocontrollers.

- Dies legt fest, welche Art der Signalwandlung durchgeführt wird.
- Abhängig hiervon liegt das Ergebnis der Signalwandlung in einem bestimmten Datentyp vor.
- Mit diesem wird innerhalb des Datenflussplans weitergerechnet.

Diese Beziehungen geben auch den Ablauf der Modellierung des mechatronischen Systems vor. Als erstes müssen die Sensoren und Aktoren ausgewählt und parametriert werden. Diese werden anschließend innerhalb der Entwicklungsumgebung mit dem virtuellen Modell des genutzten Mikrocontrollers verbunden. An dieser Stelle muss die Entwicklungsumgebung den Nutzer unterstützen, indem sie ihm zum einen die Kompatibilität der Mikrocontrolleranschlüsse zum genutzten Sensor oder Aktor anzeigt und zum anderen die Funktion des gewählten Mikrocontrollerpins entsprechend einstellt, also beispielsweise beim Anschluss eines Sensors mit analogem Signal den Analog-Digital-Wandler des Controllers vorauswählt. Nachdem die Verbindung der Sensoren und Aktoren mit dem Mikrocontrollermodell hergestellt wurde, kann der Nutzer den berechneten digitalisierten Wert innerhalb des Datenflussplans nutzen.

4.3.5 Darstellungsmöglichkeiten der Hardware

Aufgrund der geschilderten Zusammenhänge zwischen Hard- und Software bieten sich zwei Möglichkeiten der Darstellung der Hardware innerhalb der Entwicklungsumgebung an [BA2012]:

- Es kann ein separater Arbeitsbereich (zum Beispiel in einem neuen Fenster) eingerichtet werden, in welchem der Mikrocontroller dargestellt ist. In diesem Bereich kann die Hardware um den Mikrocontroller herum angeordnet und mit diesem verbunden werden. In Abhängigkeit von den angeschlossenen Sensoren und Aktoren werden innerhalb der Bibliothek des Datenflussplans die möglichen Schnittstellen zur Verfügung gestellt. Außerdem erhält der Nutzer Informationen über eventuell zu nutzende Anpassungsschaltungen. In Abbildung 40 ist eine solche mögliche Darstellung zu finden.

- Der Arbeitsbereich der Datenflussprogrammierung kann in einen Hardware-, einen Schnittstellen- und einen Softwarebereich eingeteilt werden, wobei der Hardware- und der Softwarebereich voneinander durch den Schnittstellenbereich getrennt werden. Im Hardwarebereich werden die Sensoren und Aktoren angeordnet und mit dem Schnittstellenbereich verbunden. Der Schnittstellenbereich repräsentiert die Mikrocontrollerpins und wandelt das elektrische Signal in eine Softwarerepräsentation. Im Softwarebereich kann anschließend mit diesen Daten gearbeitet werden. Zwischen der Schnittstelle und dem Hardwarebereich können Informationen bezüglich benötigter Anpassungsschaltungen zur Verfügung gestellt werden. Die hier beschriebene Darstellung wird unter Abschnitt 5.1.4 in Abbildung 49 vorgestellt.

Beide Möglichkeiten wurden anhand verschiedener Beispiele mit Hilfe des kognitiven Walkthroughs, welcher in Abschnitt 2.3.5 dargestellt ist, untersucht. Es zeigte sich, dass der erste Ansatz für Mikrocontroller mit vielen Pins besser geeignet ist, da der zweite Ansatz in diesem Fall viel Platz im Datenflussplan benötigt, um alle wichtigen Informationen darzustellen.

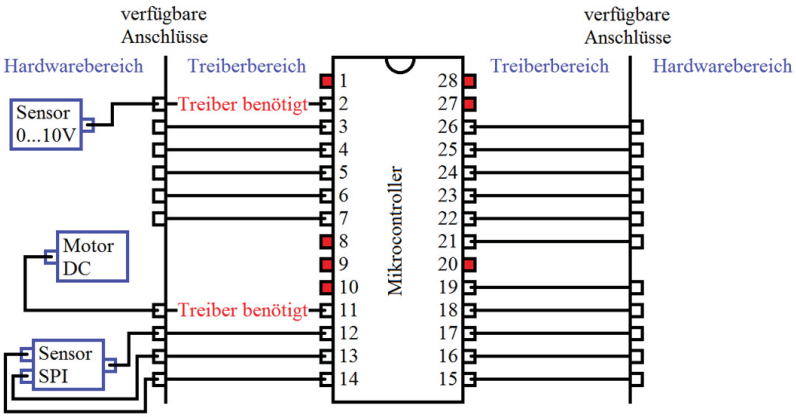

Abbildung 40: Darstellung des Mikrocontrollers, der Sensoren und der Aktoren in einem separaten Bereich innerhalb der Entwicklungsumgebung

Werden allerdings Mikrocontroller oder andere Plattformen mit wenigen Pins genutzt, ist der zweite Ansatz verständlicher, da ein direkter Bezug zwischen Hard- und Software hergestellt wird. Wird die gemeinsame Darstellung der Hardware und der Software innerhalb des Datenflussplans angestrebt, muss beachtet werden, dass die Hardware und die gewählten Schnittstellenfunktionen in allen existierenden Datenflussplänen eines Softwareprojekts gleich sein müssen, während die reine Softwarefunktionalität variabel ist.

Ein Vorteil dieser Ansätze ist, dass neben reinen Mikrocontrollern auch komplexere Plattformen dargestellt werden können. Hierfür muss lediglich das jeweilige Modell erweitert werden, wobei sich in der Darstellung im Datenflussplan eine Zweiteilung der Schnittstellenebene in eine Plattformschnittstellenebene und eine Mikrocontrollerschnittstellenebene anbietet.

Für einen Großteil der Sensoren und Aktoren, welche an Mikrocontroller oder Evaluierungsplattformen angeschlossen werden sollen, werden Schaltungen benötigt, um die jeweiligen Schnittstellen aufeinander anzupassen. Ist eine solche Anpassungsschaltung erforderlich, muss der Nutzer während der Modellierung der Hardware hierüber informiert werden. Dies muss mit Nachdruck geschehen, da ein fehlerhaftes Anschließen von Hardware den Mikrocontroller oder andere Bauteile zerstören kann. Das bedeutet, dass zumindest eine Warnmeldung erscheinen muss, welche durch einen Tastendruck zu bestätigen ist. Anschließend muss der Nutzer die Möglichkeit haben, sich über die Art des Problems zu informieren. Im besten Fall erhält der Nutzer an dieser Stelle Lösungsvorschläge, was zum Beispiel die Vorgabe nutzbarer Anpassungsschaltungen beinhalten kann. Dabei können sowohl fertige Treiber- und Anpassungsschaltungen aus dem Handel als auch einfache Schaltpläne für den Eigenbau vorgeschlagen werden.

4.4 Signale und Datentypen

Die Darstellung der Hardware innerhalb der Entwicklungsumgebung, wie sie in Abschnitt 4.3 dargestellt ist, ermöglicht dem Nutzer einen Einblick in die Signalwandlung an der Peripherie des Mikrocontrollers. Dabei werden

bestimmte Datentypen für die Darstellung unterschiedlicher Signale benötigt. In diesem Abschnitt wird daher untersucht, wie dem Nutzer die Problematik der Datentypen und der Signalarten näher gebracht werden kann.

4.4.1 Datentypen

Genau wie bei der textbasierten Programmierung werden in EasyLab Datentypen verwendet, um die Repräsentation der Daten im Mikrocontroller festzulegen. In Abschnitt 3.3.3 wurde bereits beschrieben, dass die meisten Probleme während der Nutzertests an dieser Stelle auftraten. Dabei konnte zwischen einigen Funktionsbausteinen keine Verbindung hergestellt werden, da ihre Datentypen inkompatibel waren. Dies war für den Nutzer allerdings nicht ersichtlich.

Hierfür gibt es zwei sehr gegensätzliche Lösungsansätze:

- Die Software übernimmt vollständig die Auswahl der benötigten Datentypen, so dass der Nutzer gar keinen Einfluss auf diesen Punkt nehmen muss und kann.
- Der Nutzer muss sich vollständig um die Auswahl der Datentypen kümmern.

Der erste Ansatz ist für den Nutzer zunächst der einfachste. Allerdings ist hierbei zu bedenken, dass die automatisierte Auswahl alle erdenklichen Berechnungsmöglichkeiten in Betracht ziehen muss, wodurch unweigerlich sehr mächtige Datentypen genutzt werden. Dies ist natürlich nicht ressourcenschonend, so dass der Ansatz nur für leistungsfähige Plattformen in Frage kommt.

Da im Hobbybereich aber oft mit kostengünstigen älteren Mikrocontrollern gearbeitet wird, erscheint der zweite Ansatz passender. Um diesen auch für unerfahrene Nutzer anwendbar zu gestalten, muss eine Unterstützung während der Softwareentwicklung erfolgen. Ein Umsetzungskonzept zur Lösung dieses Problems sieht vor, dass der Nutzer nicht mehr den spezifischen Datentyp auswählen muss, sondern nur die wichtigsten Eigenschaften der gewünschten Zahl.

Zur Ermittlung dieser Eigenschaften können die folgenden vier Fragen als Ausgangspunkt genutzt werden [BA2012]:

- Handelt es sich um eine binäre Entscheidung oder um eine Zahl?
- Kann die Zahl Nachkommastellen besitzen?
- Sind negative Zahlen möglich?
- Wie groß ist der mögliche Wertebereich der Zahl?

Natürlich müssen nicht immer alle Fragen beantwortet werden, da beispielsweise die Auswahl „binäre Entscheidungen“ ausreicht, um festzulegen, dass der Datentyp „boolean“ genutzt werden muss. Aus diesem Grund ist eine Realisierung in Form eines Softwareassistenten sinnvoll, bei welchem die Fragen schrittweise beantwortet werden, bis das gewünschte Zahlenformat eindeutig definiert ist.

Sobald alle benötigten Fragen beantwortet wurden, muss dem Nutzer angezeigt werden, welche Daten mit Hilfe des gewählten Datentyps realisierbar sind, so dass er dies überprüfen und bei Bedarf anpassen kann. Außerdem sollte der Name des genutzten Datentyps für den Nutzer ersichtlich sein, so dass er sich diesen einprägen kann. In Kombination mit einer Direktwahlmöglichkeit des Datentyps ergibt sich eine deutliche Erleichterung der Arbeit für erfahrene Nutzer, da sie sich nicht durch den gesamten Assistenten arbeiten müssen.

Unerfahrene Programmierer weisen eine Tendenz zur Nutzung mächtiger Datentypen auf, obwohl dies oft gar nicht notwendig ist. Auch bei der Nutzung des hier vorgestellten Konzepts ist diese Tendenz zu erwarten. Daher sollte der Nutzer auf die Konsequenzen der Nutzung mächtiger Datentypen in Bezug auf Speichereffizienz und Rechenleistung hingewiesen werden.

4.4.2 Signalarten und ihre Datentypen

In EasyLab werden im Simulationsmodus und während des Onlinedebuggings die berechneten Zahlenwerte der Ein- und Ausgänge der einzelnen Funktionsbausteine angezeigt. Dies ist in Abbildung 41 dargestellt. Das Format der Zahlen orientiert sich dabei am voreingestellten Datentyp.

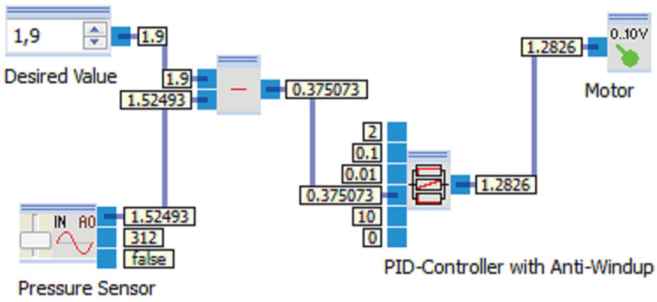

Abbildung 41: Einfaches Programm zur Druckregelung mit dargestellten Zahlenwerten an den Ein- und Ausgängen der Funktionsbausteine

Dies sollte auch im Hardwarebereich realisiert werden. Jedoch sind Rückschlüsse auf die physikalischen Prozessgrößen nicht möglich, da im unter Abschnitt 4.3 vorgestellten Konzept zur Hardwaredarstellung nur die Schnittstellen der Sensoren und Aktoren berücksichtigt werden, nicht jedoch die eigentlichen Sensoren und Aktoren mit ihrem Übertragungsverhalten. Allerdings können die elektrischen Größen an den Mikrocontrollerpins sehr einfach berechnet werden, weshalb ihre Darstellung auf der Hardwareseite der Schnittstellenebene leicht zu realisieren ist.

Auf diese Art wird dem Nutzer im Simulationsmodus und während des Onlinedebuggings die Funktionsweise der Peripherie des Mikrocontrollers anschaulich vermittelt. Die Durchführung der Simulation und des Onlinedebuggings erfolgen aber gewöhnlich erst am Ende der Modellierung der Software, so dass während dieses Schrittes noch keine Unterstützung erfolgt. Um dieses Problem zu lösen, sieht das hier vorgestellte Konzept vor, dass der Nutzer bereits während der Modellierung der Hardware über die Funktionalität der jeweiligen Peripherie informiert wird.

Wie bereits beschrieben, bestimmen die angeschlossenen Sensoren und Aktoren die Funktion dieser Peripherie, so dass beispielsweise der Anschluss eines analogen Signals bewirkt, dass der Analog-Digital-Wandler ausgewählt wird. Sobald dies geschieht, muss der Nutzer auf das Problem der Signalwandlung aufmerksam gemacht werden. Im Anschluss können weite-

re Informationen diesbezüglich abgerufen werden, so dass der Nutzer den Zusammenhang zwischen dem anliegenden elektrischen Signal und den zugehörigen Daten innerhalb des Mikrocontrollers verstehen kann.

Der Nutzer erhält somit nicht nur nützliche Informationen, welche ihm die Nutzung von Sensordaten während der Modellierung der Software erleichtern, vielmehr kann dies auch das Verständnis fördern, warum bestimmte elektrische Signalarten die Nutzung bestimmter Datentypen nach sich ziehen. In Abbildung 42 ist dieser Zusammenhang noch einmal graphisch verdeutlicht. In Abschnitt 5.1.4 wird noch einmal auf das hier vorgestellte Problem eingegangen und eine Implementierungsmöglichkeit vorgeschlagen.

4.5 Zusammenfassung

In diesem Kapitel wurden verschiedene Maßnahmen vorgestellt, welche, unter Berücksichtigung der Randbedingungen der Plattformunabhängigkeit und der Nutzbarkeit ohne domänenspezifisches Expertenwissen, die Entwicklung mikrocontrollerbasierter mechatronischer Systeme ermöglichen. Hierfür wurden einige Umsetzungskonzepte und Anforderungen vorgestellt. Diese basieren auf Ergebnissen von Tests, welche mit dem EasyKit Starter durchgeführt wurden.

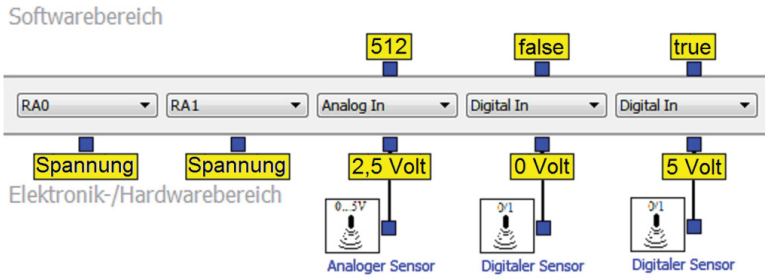

Abbildung 42 : Darstellung der Wandlung von elektrischen Signalen in ihre Repräsentation als spezifische Datentypen in der Entwicklungsumgebung

Die wichtigsten Ansätze sind:

- die Erweiterung der Programmierung auf zwei hierarchischen Ebenen um eine dritte Ebene, wobei in der ersten Ebene eine ereignisgesteuerte graphische, in der zweiten Ebene eine datenflussorientierte graphische und in der dritten Ebene eine graphisch-textuell-hybride Darstellungsform genutzt wird,
- die Möglichkeit der Modellierung der Schnittstellen der Sensoren und Aktoren in der Entwicklungsumgebung,
- die Nutzung von Softwaremodellen für die Modellierung des Verhaltens der informationstechnischen Plattform,
- die Flexibilisierung im Aufbau des Ablaufdiagramms und
- die Darstellung der Zusammenhänge zwischen den elektrischen Signalen und den im Mikrocontroller verarbeiteten Daten.

Die Konzepte sind so aufgebaut, dass sie mit geringfügigen Änderungen für unterschiedliche Plattformen eingesetzt werden können. Außerdem sind sowohl die Hardware- als auch die Softwareentwicklung sehr flexibel, so dass eine Vielzahl üblicher Funktionen aus dem Hobbybereich realisiert werden kann.

Die meisten Konzepte basieren auf der Bereitstellung von ausgewähltem Expertenwissen, welches für die Entwicklung mikrocontrollerbasierter mechatronischer System benötigt wird. Durch mehrfache Anwendung kann der Nutzer sich dieses Wissen aneignen, wodurch sein grundlegendes Verständnis für die Themengebiete der Mechanik, der Elektronik und der Informationstechnik verbessert werden kann.

5 Implementierung und Nutzertests

Im vorigen Kapitel wurden verschiedene im Rahmen dieser Arbeit entwickelte Maßnahmen für die Vereinfachung der Entwicklung mikrocontrollerbasierter mechatronischer Systeme vorgestellt. Mit Hilfe analytischer Methoden und mit Hilfe von Nutzerbefragungen wurden die vielversprechendsten Konzepte ausgewählt und beispielhaft in einer Testumgebung implementiert. Anschließend wurden mit Hilfe dieser Software Nutzertests durchgeführt. In diesem Kapitel werden zunächst die Testumgebung und anschließend die Ergebnisse der damit durchgeführten Nutzertests vorgestellt.

5.1 Implementierung in einer Testumgebung

Die Testumgebung wurde mit Hilfe von Microsoft Visual Studio [Mic2013b] und Qt [Dig2013] entwickelt. Das grundlegende Design von EasyLab wurde hierfür als Ausgangspunkt genutzt. Im Folgenden wird die Umgebung Schritt für Schritt vorgestellt. In der hier aufgeführten Reihenfolge wird auch der spätere Nutzer mit den einzelnen Bestandteilen der Oberfläche in Berührung kommen.

Um die Funktionsweise der Entwicklungsumgebung demonstrieren zu können, wurden der Arduino UNO [Ard2012a], das EasyKit Starterboard [Fes2010] und der Microchip PIC18F2520 [Mic2013a] als Beispielplattformen implementiert. Auf diese Weise sind sowohl einfache Mikrocontroller als auch Evaluierungsboards mit bereits integrierten Schaltungen abgedeckt.

Mit dem PIC18F2520 wurde bewusst eine 8-bit-Plattform gewählt. Auf diese Weise soll gezeigt werden, dass die Entwicklung mechatronischer Systeme auch mit solchen vergleichsweise ressourcenarmen Mikrocontrollern

möglich ist. Dies ist wichtig, da solche Plattformen auch heute noch im Hobbybereich regelmäßig eingesetzt werden.

Da in der Testumgebung vor allem Maßnahmen untersucht werden, welche einen Einsatz des EasyKit-Gedankens im Hobbybereich ermöglichen, erhielt die Software den Namen EasyHobby.

5.1.1 Auswahl der Hardwareplattform

Beim Starten von EasyHobby muss zunächst die gewünschte Entwicklungsplattform ausgewählt werden, da diese die mögliche Funktionalität des zu entwickelnden mechatronischen Systems beeinflusst. In Abbildung 43 ist der hierfür implementierte Dialog dargestellt.

Die Auswahl der Plattform erfolgt über eine Liste. Diese kann durch verschiedene Rahmenbedingungen eingeschränkt werden. Die Anzahl der benötigten digitalen und analogen Ein- und Ausgänge ist dabei entscheidend. Zwar sind bei der Entwicklung mikrocontrollerbasierter mechatronischer Systeme oft auch andere Eigenschaften wichtig, jedoch sollen hier nur die wichtigsten berücksichtigt werden. Auf diese Weise sind eine gute Übersichtlichkeit und eine leichte Bedienbarkeit gewährleistet. Zur weiteren Verbesserung der Verständlichkeit wurden Beispiele für die einzelnen Signalarten angegeben. Auf eine weiterführende Erklärung wurde an dieser Stelle verzichtet, um zu überprüfen, ob die vorgegebenen Informationen bereits ausreichend für die Testnutzer sind.

Nach der Auswahl der gewünschten Plattform erhält der Nutzer Informationen über ihre Inbetriebnahme. Dieser Informationsdialog ist für den PIC18F2520 in Abbildung 44 dargestellt.

Der Dialog beinhaltet zwei Schaltpläne. Der obere Plan wird zur Programmierung benötigt, während der untere Plan genutzt werden kann, um den bereits programmierten Controller zu betreiben. Zwar sind auch andere Beschaltungen des Mikrocontrollers denkbar, wie zum Beispiel die Nutzung eines Oszillators mit einer anderen Taktfrequenz, allerdings sollen dem Nutzer aus Gründen der Verständlichkeit nur diese zwei Schaltungen vorgestellt werden.

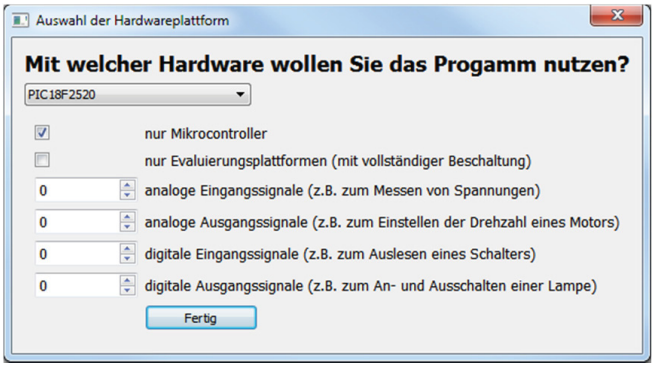

Abbildung 43: Dialog zur Auswahl der Hardwareplattform

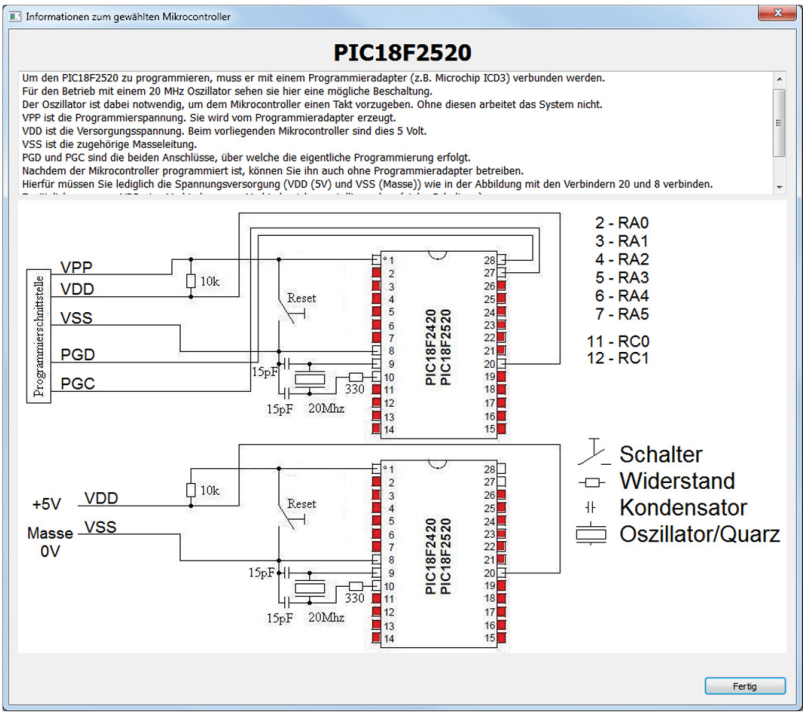

Abbildung 44: Informationsdialog zur Inbetriebnahme der Plattform

Sämtliche in den Schaltungen vorhandenen Bestandteile sind noch einmal separat aufgeführt und mit Namen bezeichnet, so dass der Nutzer diese im Schaltplan zuordnen kann. Auch die verschiedenen Mikrocontrollerpins sind noch einmal mit den Bezeichnungen aufgeführt, mit welchen sie auch später während der Programmierung im Datenflussplan bezeichnet sind.

Im oberen Bereich befindet sich eine umfassende textuelle Beschreibung der Eigenschaften der Schaltung. Hierzu gehören beispielsweise Hinweise zur Spannungsversorgung, zur Nutzung des Programmieradapters und auch zu Bezugsquellen benötigter elektronischer Bauteile.

Der Informationsdialog kann über das Menü der Hauptarbeitsoberfläche jederzeit wieder angezeigt werden, so dass der Nutzer auch während der Programmierung noch Zugriff auf diese Informationen hat.

5.1.2 Hauptarbeitsoberfläche

In der Hauptarbeitsoberfläche von EasyHobby, welche in Abbildung 45 dargestellt ist, befinden sich alle zur Modellierung des mechatronischen Systems benötigten Komponenten. Dies beinhaltet sowohl Werkzeuge für die Modellierung der Sensoren- und Aktorenschnittstellen als auch für die Programmierung der eigentlichen Mikrocontrollersoftware. Die Bestandteile der Hauptarbeitsoberfläche werden im Folgenden kurz beschrieben.

Menü und allgemeine Werkzeugleiste

Wie in den meisten anderen Anwendungsprogrammen werden auch in EasyHobby ein Menü und eine Werkzeugleiste benötigt. Diese ist im oberen Bereich der Oberfläche zu finden. Neben den üblichen Bestandteilen zum Laden und Speichern von Projekten befinden sich hier vor allem Werkzeuge, welche zur Manipulation der Softwaremodelle genutzt werden können.

Logbuch

Das Logbuch ist ebenso in vielen Anwendungsprogrammen, vor allem aber in fast allen Entwicklungsumgebungen zu finden. In Abbildung 45 ist dieses im unteren Bereich des Fensters enthalten. Die Hauptaufgabe ist die Ausgabe von Statusinformationen an den Nutzer.

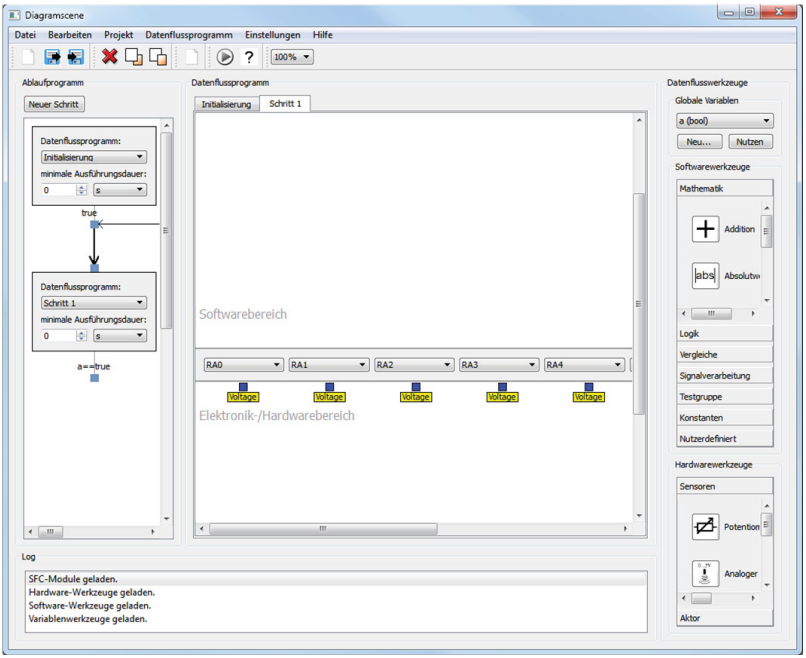

Abbildung 45: Hauptarbeitsoberfläche der Testumgebung

Ablaufdiagramm

In Abbildung 45 ist auf der linken Seite das Ablaufdiagramm zu finden, in welchem das Hauptprogramm mit Hilfe der Ablaufsprache modelliert wird. In Abbildung 46 ist ein solches Ablaufdiagramm noch einmal vollständig dargestellt.

Um eine bessere Übersichtlichkeit zu ermöglichen, kann im Ablaufdiagramm gezoomt werden. Auf diese Weise sind auch komplexere Diagramme realisierbar, ohne dass die Darstellung zu viel Platz in der Oberfläche einnimmt. So ist es möglich, dass das Ablaufdiagramm jederzeit im Vordergrund dargestellt bleiben kann, wie dies in Abschnitt 4.2 gefordert wurde.

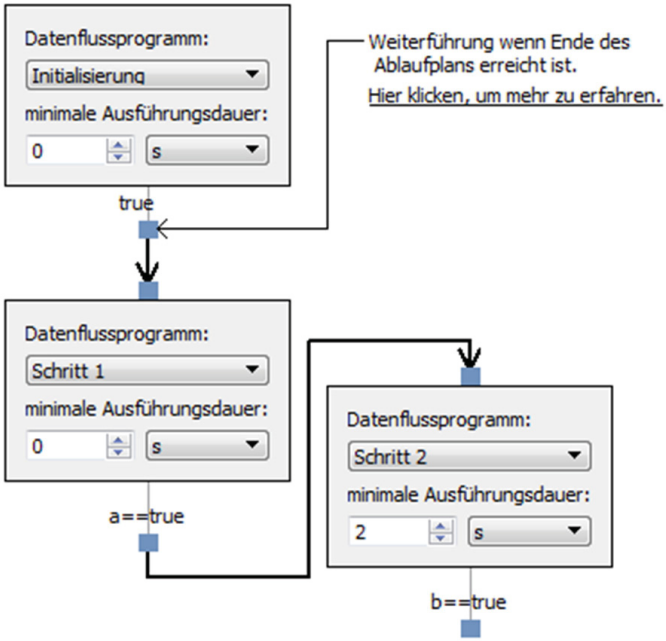

Abbildung 46: Ablaufdiagramm in der Testumgebung

Ein bestehendes Ablaufdiagramm kann erweitert werden, indem ein neuer Zustand hinzugefügt und in gewünschter Weise mit dem bereits vorhandenen Diagramm verbunden wird. Jeder Zustand besitzt einen Eingangsverbinder und ein bis vier Ausgangsverbinder. Jedem Zustand ist ein Unterprogramm in Form eines Datenflussplans zugeordnet.

Die Verbindungen zwischen den Ein- und Ausgangsverbindern der Zustände bestimmen den Programmablauf. Jedem Ausgangsverbinder ist dabei eine Bedingung zugeordnet. Ist eine Bedingung an einem Ausgang erfüllt, so wird der an diesem Ausgang verbundene Zustand als nächstes ausgeführt. Für jeden Zustand kann festgelegt werden, wie lang die Mindestausführungsdauer für eine Ausführung des Datenflussplans sein soll. Die Ausgangsverbinder werden über einen separaten Dialog parametriert, welcher in Abbildung 47 dargestellt ist. Dabei wird für die Festlegung der Bedingung

entweder eine bereits existierende Variable genutzt oder es wird eine neue Variable angelegt, welche später im Datenflussplan eingesetzt werden kann.

Im oberen rechten Bereich des Ablaufdiagramms in Abbildung 46 befindet sich ein Hinweis, welcher dem Nutzer erklärt, dass beim Erreichen des Endes des Ablaufdiagramms die Ausführung wieder direkt nach dem Initialisierungszustand beginnt. Im Ablaufdiagramm, welches in Abbildung 46 dargestellt ist, bedeutet dies, dass das Programm wieder in den Datenflussplan „Schritt 1“ springt, nachdem „Schritt 2“ ausgeführt wurde und die Bedingung „b==true“ erfüllt ist.

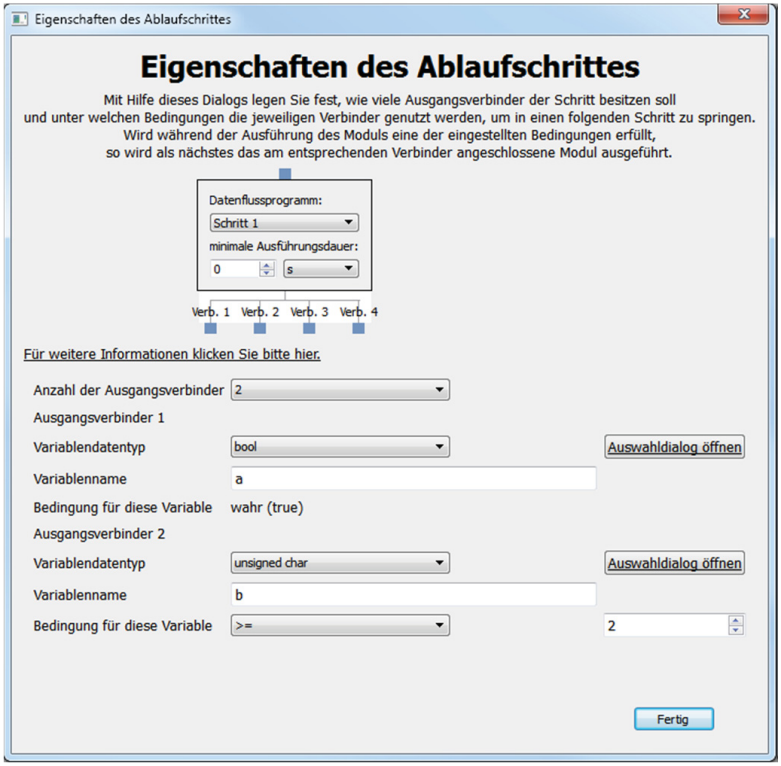

Abbildung 47: Dialog zur Parametrierung der Ausgangsverbinder von Zuständen im Ablaufdiagramm

Datenflussplan in der Funktionsbausteinsprache

Im Zentrum der in Abbildung 45 dargestellten Hauptarbeitsoberfläche befindet sich die Oberfläche zur Modellierung der Datenflusspläne. Wie bereits beschrieben, repräsentieren diese die Funktionalität der Unterprogramme, welche den einzelnen Zuständen des Ablaufdiagramms zugeordnet werden.

In Abschnitt 4.3.5 wurden zwei Möglichkeiten zur Darstellung der Hardware innerhalb der Entwicklungsumgebung beschrieben. In EasyHobby wurde eine Variante realisiert, bei welcher diese Darstellung innerhalb des Datenflussplans erfolgt. Hierfür wurde dieser in drei Bereiche aufgeteilt. Im unteren Bereich erfolgt die Modellierung der elektronischen Hardware, welche an den Mikrocontroller oder die Evaluierungsplattform angeschlossen wird. Im oberen Bereich wird die Software modelliert, welche später als Unterprogramm auf den Mikrocontroller gebrannt wird. Dazwischen befindet sich die Schnittstellenebene, auf welcher die Wandlung zwischen den elektrischen Signalen und den Mikrocontrollerdaten innerhalb der Software erfolgt.

Werkzeuge für die Modellierung von Datenflussplänen

Die Datenflusspläne werden mit Hilfe von Funktionsbausteinen modelliert. Diese befinden sich in einem Werkzeugbereich auf der rechten Seite der in Abbildung 45 dargestellten Hauptarbeitsoberfläche. Sie können in drei Gruppen eingeteilt werden:

- Hardwarefunktionsbausteine, mit welchen Sensoren und Aktoren im Hardwarebereich des Datenflussplans modelliert werden können,
- Softwarefunktionsbausteine, mit welchen in erster Linie Berechnungen, Vergleiche und logische Operationen im Softwarebereich des Datenflussplans realisiert werden können und
- Variablenfunktionsbausteine, welche innerhalb des Softwarebereichs des Datenflussplans für die Zwischenspeicherung und das Auslesen globaler Variablen genutzt werden können.

5.1.3 Modellierung der Sensoren und Aktoren

Um die Sensoren und Aktoren im Datenflussplan modellieren zu können, müssen die gewünschten Hardwarefunktionsbausteine aus dem Werkzeugbereich in den Hardwarebereich des Datenflussplans eingefügt werden. Anschließend müssen diese mit den entsprechenden Mikrocontrollerpins der Schnittstellenebene verbunden werden, wobei die Verbindungen Signalflüsse darstellen, keine Schaltpläne.

In Abschnitt 4.3.3.3 wurde bereits beschrieben, dass die Hardwarebibliothek nicht jeden möglichen Sensor und Aktor beinhalten kann, da dies nicht realistisch handhabbar ist. Vielmehr müssen sinnvolle Gruppen ähnlicher Hardware gebildet werden, welche sich lediglich in einigen Parametern unterscheiden. Dies wurde in EasyHobby berücksichtigt. So existiert beispielsweise für unidirektionale Gleichstrommotoren lediglich ein einziger Hardwarefunktionsbaustein. Dieses muss nach dem Einfügen in den Datenflussplan mit Hilfe des in Abbildung 48 dargestellten Dialogs vom Nutzer parametriert werden.

Grundsätzlich werden für Aktoren die drei Parameter Signalart, Spannung sowie Strom, und für Sensoren die zwei Parameter Signalart und Signalwertebereich abgefragt. Diese Parameter werden benötigt, um die jeweils zu integrierende Treiberschaltung für den Anschluss der Sensoren und Aktoren zu ermitteln. Außerdem wird mit Hilfe der Signalart bestimmt, welche Mikrocontrollerpins oder Anschlüsse des Evaluierungsboards für die Verbindung mit den Sensoren und Aktoren grundsätzlich in Frage kommen.

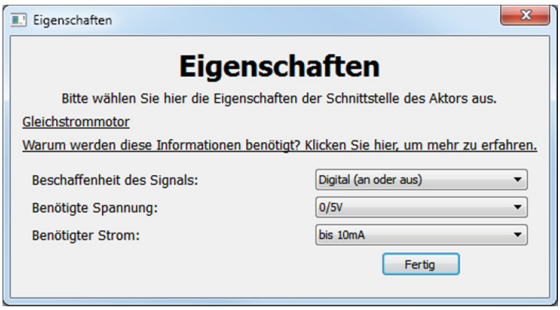

Abbildung 48: Dialog zur Parametrierung der Hardwarefunktionsbausteine

Um dies realisieren zu können, müssen für alle sinnvollen Kombinationen der Parameter die folgenden Informationen hinterlegt sein:

- Wird überhaupt eine Treiber- oder Anpassungsschaltung benötigt?
- Wenn ja, muss eine beispielhafte Schaltung als Graphik mit zusätzlichen Informationen in textueller Form vorgegeben sein.
- Welche Signalart kann mit welchem Typ Mikrocontrollerpin verbunden werden?

In EasyHobby wurden diese Informationen in XML-Dateien hinterlegt, wodurch die Bearbeitung und Erweiterung auch ohne Veränderung des Quellcodes möglich ist.

Die Anzahl der Parameterkombinationen variiert stark. Sie ist abhängig davon, wie viele unterschiedliche Treiber- und Anpassungsschaltungen für jede sinnvolle Kombination existieren. Meist ist die Anzahl der sinnvollen Kombinationen jedoch relativ gering. In vielen Fällen beschränkt sich die Auswahl der Signalart auf einen einzigen Punkt, da beispielsweise ein Schalter nur ein binäres Signal erzeugen kann. Auch bei den möglichen Spannungen und Stromstärken sind einige wenige Auswahlpunkte ausreichend, da die meisten Grundschaltungen für unterschiedliche Spannungs- und Strombereiche genutzt werden können. Diese Schaltungen unterscheiden sich oft nur in einigen Widerstandswerten, welche vom Nutzer selbst berechnet werden müssen. Hierfür wird in der textuellen Beschreibung erklärt, wie die Dimensionierung der Widerstände erfolgt. So wird beispielweise für analoge Sensorschnittstellen ein allgemeiner Spannungsteiler vorgeschlagen, für welchen der Nutzer über vorgegebene Gleichungen die Widerstandswerte berechnen muss.

5.1.4 Funktion der Schnittstellenebene

Die Schnittstellenebene visualisiert die Wandlung zwischen dem elektrischen Signal des Hardwarebereichs und der Datentyprepräsentation im Softwarebereich. Jeder einzelne Anschluss des gewählten Mikrocontrollers oder der gewählten Evaluierungsplattform sind hier dargestellt. In XML-Dateien sind Informationen über die Peripherie der jeweiligen Anschlüsse hinterlegt. Von EasyHobby werden Funktionen als digitale Ein-/Ausgänge,

Analog-Digital-Umwandler, pulsweitenmodulierte Ausgangssignale, analoge Ausgangssignale und BUS-Schnittstellen unterstützt. Im Grundzustand sind die Anschlüsse der Plattform noch funktionslos. Sie besitzen einen Verbinder in Richtung des Hardwarebereichs, jedoch keinen in Richtung des Softwarebereichs des Datenflussplans. Wie in Abbildung 49 dargestellt, wird in diesem Fall der Name des jeweiligen Anschlusses angezeigt.

Sobald Hardware im Hardwarebereich eingefügt und parametriert wurde, kann sie an einen Verbinder angeschlossen werden. Während des Herstellens der Verbindung markiert EasyHobby alle verfügbaren Verbinder entsprechend ihrer Kompatibilität nach dem in Tabelle 4 aufgeführten Schema.

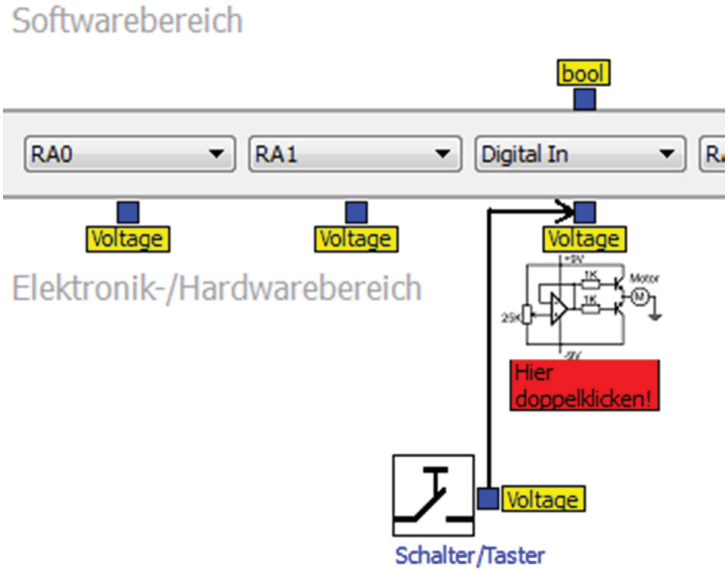

Abbildung 49: Schnittstellenbereich des Datenflussplans mit einem angeschlossenem Schalter und zwei nicht belegten Anschlüssen

Farbe	Eigenschaft
Grün	Die Verbindung ist möglich. Es werden keine Treiber oder Anpassungsschaltungen für eine Verbindung benötigt.
Gelb	Die Verbindung ist möglich. Allerdings werden Treiber oder Anpassungsschaltungen für eine Verbindung benötigt.
Rot	Die Verbindung ist nicht möglich.

Tabelle 4: Farbkodierung der Kompatibilität von elektrischem Signal und Plattformperipherie

Für diese Markierung werden die Parameter der eingefügten Hardware und die Informationen über die Peripherie der Plattform genutzt. Dabei wurden allerdings nur Anpassungsschaltungen berücksichtigt, welche mit vertretbarem Aufwand realisierbar sind und im Hobbybereich sinnvoll einsetzbar sind. So wurde beispielsweise bewusst nicht berücksichtigt, dass ein digitaler Ein-/Ausgang durch die Nutzung eines zusätzlichen Analog-Digital-Wandlers auch analoge Signale erfassen kann. Dies wäre vergleichsweise kompliziert und gewöhnlich auch unnötig, da selbst im Hobbybereich eingesetzte, kostengünstige Plattformen im Normalfall mehrere in der Peripherie integrierte Analog-Digital-Wandler zur Verfügung stellen.

Ist die Verbindung zwischen dem Hardwarefunktionsbaustein und dem Verbinder der Schnittstelle nur mit Hilfe einer Anpassungsschaltung möglich, so informiert EasyHobby den Nutzer hierüber mit Nachdruck, sobald die Verbindung hergestellt wird. Dies ist wichtig, da ein fehlerhaftes Anschließen von Hardware zur Zerstörung des Systems führen kann. Es wird ein Warndialog angezeigt und der betroffene Verbinder der Schnittstelle wird, wie in Abbildung 49 dargestellt, markiert.

Durch Doppelklicken auf die Markierung kann sich der Nutzer über das bestehende Problem informieren. Hierfür werden ihm die Informationen dargestellt, welche für die zuvor festgelegte Parameterkombination der Hardware hinterlegt wurden. Ein Beispiel hierfür ist in Abbildung 50 zu finden.

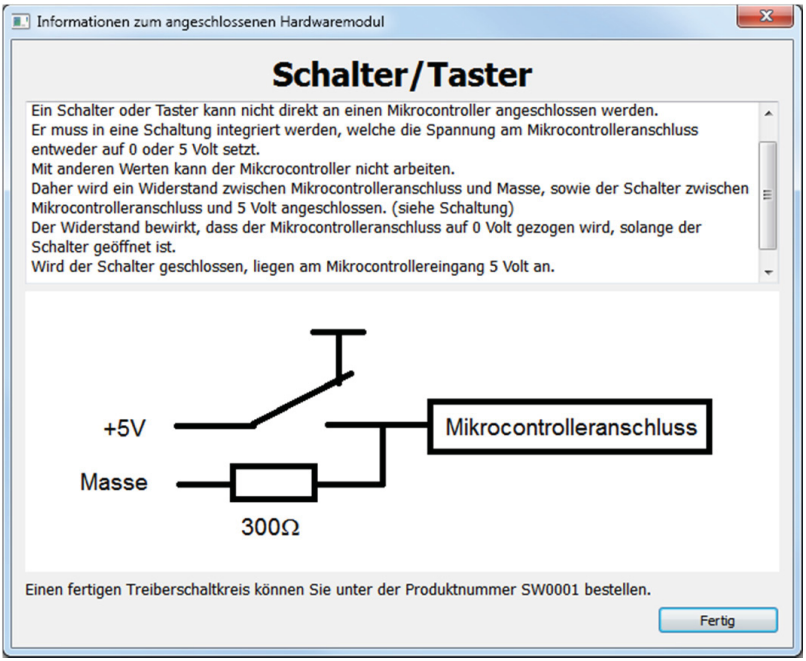

Abbildung 50: Dialog zur Darstellung von Informationen über benötigte Schaltungen

In der Abbildung ist zu erkennen, dass unter der Darstellung der Schaltung auch ein Bereich vorgesehen ist, in welchem eine vorgefertigte Treiberschaltung vorgeschlagen wird.

Nachdem die Verbindung zwischen Hardware und Plattformanschluss hergestellt wurde, wird die Funktion des Anschlusses automatisch eingestellt. Soll also beispielsweise ein analoges Signal verarbeitet werden, so wird der Analog-Digital-Wandler ausgewählt. Außerdem wird ein neuer Verbinder im Softwarebereich eingefügt, welcher das elektrische Signal am Plattformanschluss repräsentiert. Der Datentyp dieses Verbinders wird auf das am Verbinder des Hardwarebereichs anliegende elektrische Signal abgestimmt. Mit diesen Daten kann anschließend im Softwarebereich des Datenflussplans gearbeitet werden.

Wie bereits in Abschnitt 4.4.2 beschrieben, kann der Nutzer die Zusammenhänge zwischen dem elektrischen Signal im Hardwarebereich und der zugehörigen Repräsentation im Softwarebereich nur im Simulationsmodus und während des Online-Debuggings erkennen. Zum besseren Verständnis dieser Problematik wurde ein weiterer Informationsdialog implementiert, welcher dem Nutzer angezeigt wird, sobald er einen Plattformanschluss mit einem Signal verbindet. Je nach Signalart und Signalrichtung wird in diesem Dialog in vereinfachter Form beschrieben, wie aus einem elektrischen Signal ein bestimmter Datentyp oder aus einem bestimmten Datentyp ein elektrisches Signal erzeugt wird. In Abbildung 51 ist ein solcher Dialog dargestellt.

5.1.5 Behandlung von Datentypen

In EasyHobby soll die Datentypauswahl vom Nutzer durchgeführt werden. Hierfür wurde ein Dialog zur Unterstützung implementiert. Dieser ist in Abbildung 52 dargestellt. Er wird aufgerufen, sobald ein neuer Softwarefunktionsbaustein in den Datenflussplan eingefügt wird.

Auf der linken Seite des Dialogs befinden sich die Fragestellungen, durch deren Beantwortung auf der rechten Seite der gewünschte Datentyp automatisch eingestellt wird. Zusätzlich wird der darstellbare Wertebereich angezeigt. An dieser Stelle wurde aus Gründen der Übersichtlichkeit darauf verzichtet, dem Nutzer Hinweise zum Ressourcenbedarf der Datentypen zu geben. Diese Hinweise befinden sich auf einer zusätzlich Hilfeseite, welche aus dem Dialog heraus aufgerufen werden kann.

Fortgeschrittene Nutzer können den Datentyp direkt auswählen und somit Zeit sparen. In EasyHobby werden typische Datentypen der Sprache C genutzt. Bei häufiger Nutzung der Software kann sich der Nutzer diese Datentypen einprägen, so dass der spätere Einstieg in textuelle Programmiersprachen erleichtert werden kann.

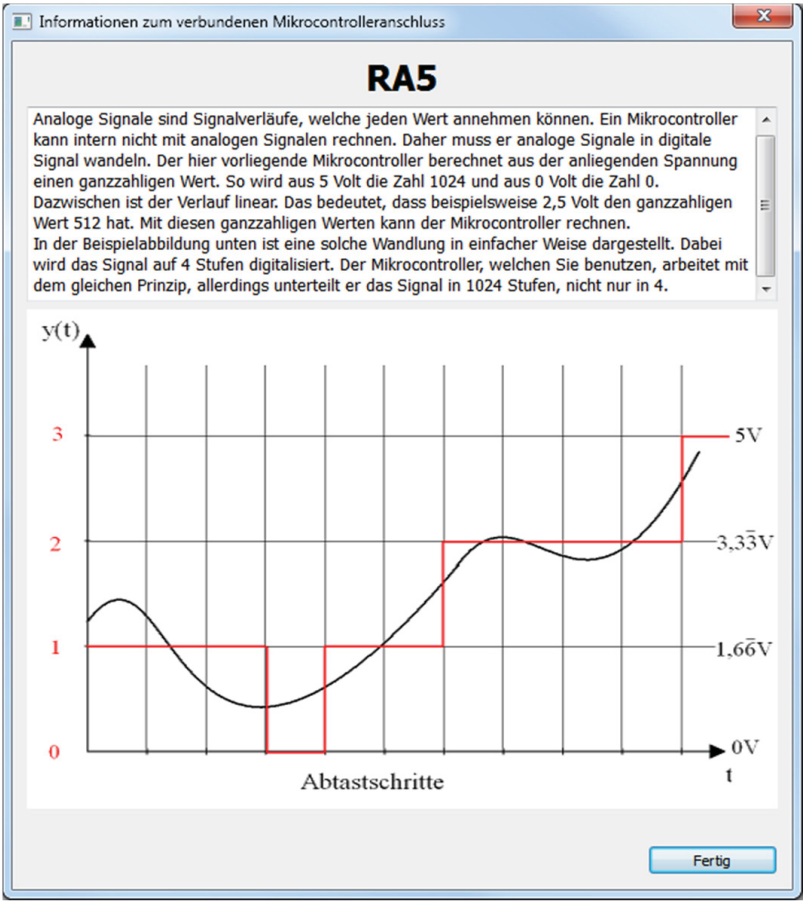

Abbildung 51: Informationsdialog zur Wandlung elektrischer Signale in bestimmte Datentypen (hier: Beispielhafte Wandlung eines analogen Signals in einen Integer-Wert am Analog-Digital-Wandler von Mikrocontrolleranschluss RA5)

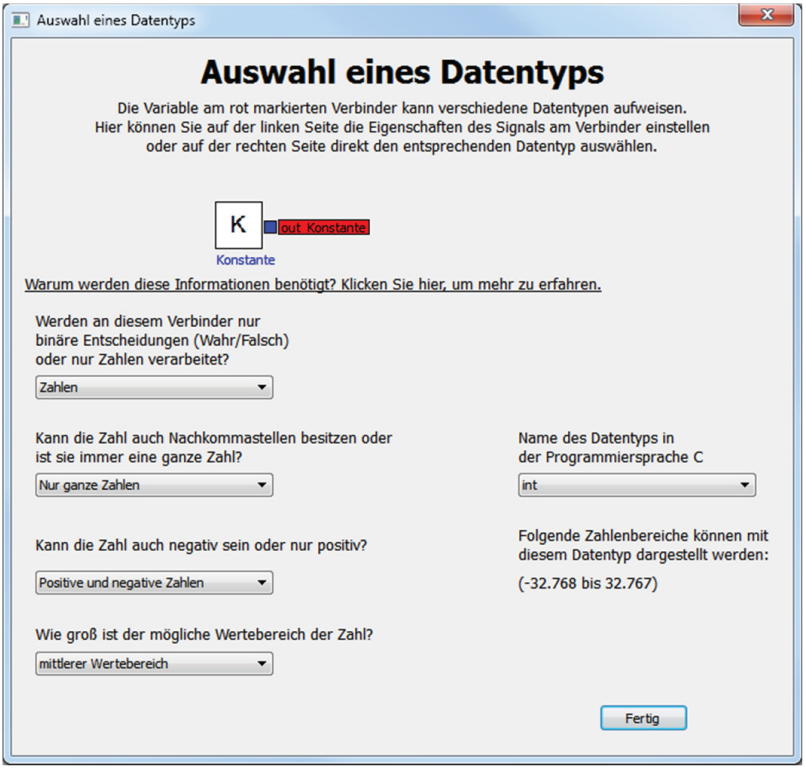

Abbildung 52: Dialog zur Auswahl des gewünschten Datentyps

5.1.6 Parametrierung der Funktionsbausteine des Datenflussplans

Die Funktionsbausteine, mit welchen die Softwarefunktionalität im Datenflussplan modelliert wird, müssen im Normalfall nicht separat konfiguriert werden. Die einzigen Parameter sind für jeden Eingangs- und Ausgangsverbinder der Datentyp und der Standardwert. Die Datentypen werden gewöhnlich bereits beim Einfügen des Funktionsbausteins in den Datenflussplan festgelegt, wie dies in Abschnitt 5.1.5 beschrieben wurde.

Die Standardwerte an den Eingangsverbindern werden gewöhnlich überschrieben, sobald eine Verbindung zu einem Ausgangsverbinder eines anderen Funktionsbausteins hergestellt wird. Der Standardwert am Ausgang wird im Allgemeinen durch das Ergebnis der Berechnung überschrieben. Dennoch existieren verschiedene Situationen, in welchen die Parametrierung eines Softwarefunktionsbausteins sinnvoll ist. So kann beispielsweise einem Eingangsverbinder ein konstanter Wert vorgegeben sein. In diesem Fall muss dieser Wert als Standardwert eingetragen und der gewünschte Datentyp gewählt werden. Für die Festlegung der Datentypen kann aus dem in Abbildung 53 vorgestellten Dialog auch der in Abschnitt 5.1.5 vorgestellte Dialog zur Datentypauswahl aufgerufen werden. Innerhalb des Dialogs zur Parametrierung der Softwarefunktionsbausteine ist aus Gründen der Übersichtlichkeit keine nähere Beschreibung des Funktionsbausteins dargestellt. Diese kann aber aus dem Dialog heraus aufgerufen werden.

5.1.7 Implementierung des Drei-Ebenen-Konzepts

Um einen neuen Softwarefunktionsbaustein zu erstellen, sind grundsätzlich erweiterte Fähigkeiten notwendig, so dass der in diesem Abschnitt vorgestellte Ansatz nicht für die Nutzung durch Einsteiger gedacht ist. Dennoch soll die Entwicklung möglichst einfach sein, so dass keine tiefgehenden Programmierkenntnisse oder sogar das selbstständige Programmieren in einer textuellen Programmiersprache nötig sind.

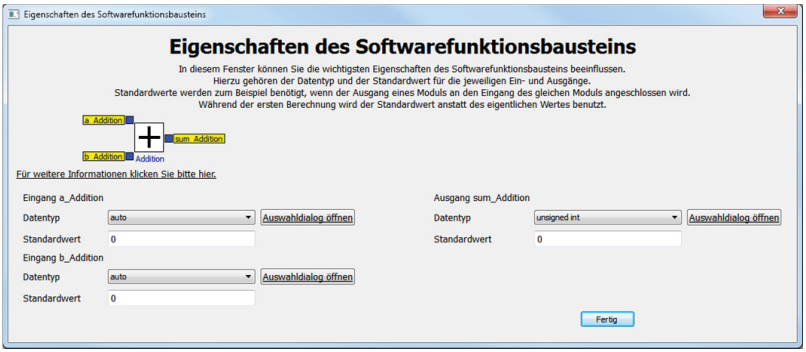

Abbildung 53: Dialog zur Parametrierung der Softwarefunktionsbausteine

Das in Abschnitt 4.1 vorgestellte Konzept zur Erstellung benutzerdefinierter Softwarefunktionsbausteine sieht eine Kombination aus graphischen und textuellen Elementen zur Programmierung vor. In Abbildung 54 ist die auf diesem Konzept basierende und in EasyHobby implementierte Entwicklungsoberfläche dargestellt.

Auf der rechten Seite befindet sich die Liste der Softwarefunktionsbausteine, wie sie bereits aus dem Werkzeugbereich des Datenflussplans bekannt sind. Diese wurde durch eine bedingte Verzweigung (if-Anweisung) und eine Wiederholung (for-Schleife) erweitert. Links befinden sich im oberen Bereich die Bearbeitungs- und Parametrierungswerkzeuge und im unteren Bereich der Quellcodeeditor.

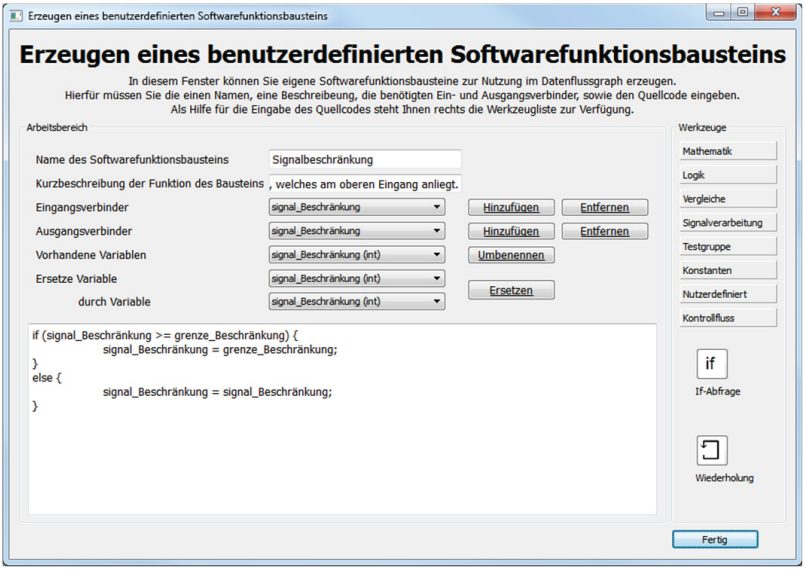

Abbildung 54: Entwicklungsoberfläche für ein hybrides Konzept zum Erzeugen von benutzerdefinierten Funktionsbausteinen auf der dritten Ebene der Programmierung, bestehend aus graphischen und textuellen Elementen

Sobald der Nutzer einen Funktionsbaustein aus der Liste auf der rechten Seite auswählt, wird dessen Quellcode in den Quellcodeeditor geschrieben. Funktionsbausteine, welche nicht intuitiv verständlich sind, werden mit Hilfe von Kommentaren erklärt. Einfachere Funktionsbausteine, wie zum Beispiel Grundrechenarten, werden aus Gründen der Übersichtlichkeit nicht erklärt.

Die im eingefügten Funktionsbaustein existierenden Variablen stehen im Werkzeugbereich des Dialogs zur weiteren Verarbeitung zur Verfügung. Sie können mit Hilfe dieser Werkzeuge umbenannt, ersetzt oder den Ein- und Ausgangsverbindern des zu erstellenden Funktionsbausteins zugeordnet werden. Der Nutzer kann diese Werkzeuge einsetzen. Er kann die Bearbeitung jedoch auch vollständig von Hand durchführen.

Eine Gültigkeitsprüfung des Quellcodes wurde nicht implementiert, da EasyHobby in erster Linie für Nutzertests entwickelt wurde. Auch weitere Werkzeuge, wie die Möglichkeit zur Verlinkung externer Bibliotheken oder die assistentenunterstützte Erstellung von Quellcode zur Abfrage einfacher serieller Datenverbindungen, wurden aus diesem Grund nicht implementiert.

Die neu erstellten Funktionsbausteine werden in eine separate Gruppe eingeordnet, so dass diese leicht identifiziert werden können. Ansonsten sind sie aber genau wie die standardmäßig vorhandenen Funktionsbausteine nutzbar. Das bedeutet, dass sie auch für die Erstellung weiterer benutzerdefinierter Funktionsbausteine eingesetzt werden können.

Einmal erstellte benutzerdefinierte Funktionsbausteine können problemlos später bearbeitet und entfernt werden. Für die Bearbeitung wird, genauso wie bei der Erstellung, der in Abbildung 54 dargestellte Dialog genutzt.

5.1.8 Vorgehen bei der Entwicklung eines mikrocontrollerbasierten mechatronischen Systems mit Hilfe der Testumgebung

In diesem Abschnitt wird beschrieben, wie ein einfaches mikrocontrollerbasiertes mechatronisches System mit Hilfe der Testentwicklungsumgebung EasyHobby entwickelt werden kann. Im Beispiel wird ein Gleichstrommotor

durch einen Taster für zwei Sekunden eingeschaltet. Als Plattform wird der PIC18F2520 Mikrocontroller der Firma Microchip [Mic2013a] genutzt.

Zunächst startet der Nutzer die Software und wählt mit Hilfe des in Abbildung 43 dargestellten Dialogs den PIC18F2520 als Entwicklungsplattform aus. Anschließend wird der Informationsdialog zur Inbetriebnahme des Mikrocontrollers angezeigt. Der Nutzer kann nun den Mikrocontroller zur Programmierung vorbereiten. Dieser Schritt kann allerdings auch auf einen späteren Zeitpunkt verschoben werden.

Nun erfolgt die Modellierung des Ablaufdiagramms für den Mikrocontroller. Im vorgegebenen Beispiel müssen zwei neue Zustände eingefügt und mit je einem Datenflussplan verbunden werden. Nach der Initialisierung soll in den ersten Zustand gewechselt werden. Durch einen Tastendruck soll aus dem ersten Zustand in den zweiten gewechselt werden. Nach zwei Sekunden soll wieder in den ersten Zustand gewechselt werden. Jedem der beiden Zustände wird beim Einfügen in das Ablaufdiagramm eine Variable des Typs „bool“ zugeordnet. Außerdem wird die minimale Ausführungsdauer des zweiten Zustands auf 2 Sekunden eingestellt. Das somit entstehende Ablaufdiagramm ist in Abbildung 46 dargestellt.

Nachdem das Ablaufdiagramm fertiggestellt wurde, müssen die beiden Datenflusspläne erstellt werden. Dabei wird nicht nur die Softwarefunktionalität, sondern die Gesamtfunktion des mechatronischen Systems modelliert. Als erstes werden der Taster und der Gleichstrommotor in den Hardwarebereich eingefügt und an die gewünschten Verbinder angeschlossen. Es ist egal, in welchem Datenflussplan dies erfolgt, da die Hardware identisch in allen Plänen dargestellt wird.

Im Dialog zur Parametrierung der Hardware (siehe Abbildung 48) müssen für den Taster keine Anpassungen vorgenommen werden. Für den Gleichstrommotor müssen der Spannungsbereich und die Stromaufnahme eingestellt werden. Sowohl für den Taster als auch für den Motor sind Anpassungsschaltungen notwendig, welche sich der Nutzer anzeigen lassen und zum Anschließen der Hardware an den Controller nutzen kann.

Nun erfolgt die Modellierung der Software innerhalb der Datenflusspläne. Im Datenflussplan, welcher dem ersten Zustand zugeordnet ist, soll der Motor ausgeschaltet sein. Daher wird eine Konstante eingefügt, als Datentyp

„bool“ festgelegt und mit dem Mikrocontrolleranschluss, an welchem der Motor angeschlossen ist, verbunden. Der Standardwert der Konstante muss nun noch auf „false“ gesetzt werden. Durch Betätigen des Tasters soll das Programm in den nächsten Zustand springen. Das bedeutet, dass die Variable des Typs „bool“ am Ausgang des ersten Zustands den Wert „true“ haben muss. Folglich muss diese Variable lediglich in den Softwarebereich des Datenflussplans eingefügt und mit dem Taster verbunden werden. Wird der Taster gedrückt, so ändert sich der Wert der Variable auf „true“ und die Bedingung am Ausgang des Zustands im Ablaufdiagramm ist erfüllt. Im zweiten Zustand ist der Aufbau ähnlich. An den Mikrocontrolleranschluss mit dem Motor wird nun eine Konstante mit dem Wert „true“ angeschlossen.

Der zweite Zustand soll nur einmal durchlaufen werden. Anschließend soll sofort in den ersten Zustand zurück gewechselt werden. Daher muss die Variable am Ausgang des zweiten Zustands des Ablaufdiagramms auf „true“ gesetzt werden. Dies kann über das Einstellen des Standardwertes der Variable oder über eine vorgeschaltete Konstante erfolgen. In Abbildung 55 sind die beiden Datenflusspläne dargestellt.

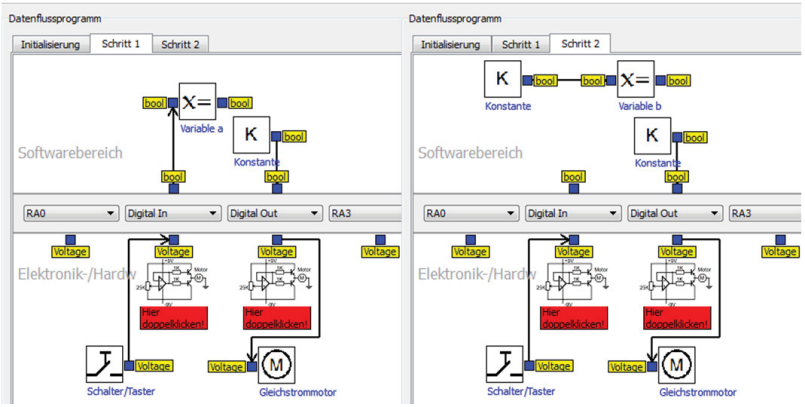

Abbildung 55: Datenflusspläne des Beispielprogramms zum tasteraktivierten, zeitgesteuerten Leuchten einer lichtemittierenden Diode

Das mechatronische System ist nun modelliert. Mit Hilfe der Informationen aus dem Datenflussplan und aus der Schaltung zur Inbetriebnahme der Plattform kann der Nutzer die Hardware zusammenstellen und aufbauen. Außerdem kann aus dem Modell heraus die Software für den Mikrocontroller erzeugt werden.

5.2 Nutzertests

Die in Abschnitt 5.1 vorgestellte Software wurde gemeinsam mit Personen der möglichen Nutzergruppe getestet. Zentrale Fragestellung dabei war, ob die Nutzer mit Hilfe der Testumgebung in der Lage sind, ein funktionsfähiges mikrocontrollerbasiertes mechatronisches System selbstständig zu entwickeln. Vor allem wird der Einfluss der im vorigen Kapitel vorgestellten und implementierten Ansätze untersucht.

Im Einzelnen sollte geklärt werden, ob Nutzer ohne domänenspezifisches Expertenwissen mit Hilfe der Testumgebung

- eine Schaltung zum Programmieren und Betreiben eines Mikrocontrollers aufbauen,
- die Sensoren und Aktoren mit den entsprechenden benötigten Schaltungen anschließen und
- die Software für den Mikrocontroller modellieren können.

Außerdem sollten die folgenden Fragen untersucht werden:

- Verstehen die Nutzer, während der Nutzung der Testumgebung, einige Grundlagen der Elektronik- und Softwareentwicklung sowie der Signalverarbeitung?
- Weckt die Nutzung der Software Interesse an mechatronischen Systemen?
- Trauen sich die Nutzer zu, das System auch allein einzusetzen?

5.2.1 Auswahl der Testnutzer

Grundsätzlich besteht immer das Ziel, dass die ausgewählten Testnutzer die eigentliche Nutzergruppe möglichst gut repräsentieren. Mit der hier vorgestellten Software sollen Nutzer ohne domänenspezifisches Expertenwissen ein technisches Problem lösen können. Sie sollen bereits ein grundsätzliches technisches Interesse mitbringen und nach Möglichkeit ein technisches Hobby ausüben.

Personen mit einem solchen Hobby beschäftigen sich allerdings beim Auftreten eines technischen Problems gewöhnlich selbstständig mit einer Lösungsfindung, wodurch Expertenwissen auf dem jeweiligen Gebiet erlangt wird. Nutzer mit Expertenwissen sind jedoch für den Test ungeeignet. Aus diesem Grund wurden mögliche Nutzer gesucht, welche bereits ausreichend Wissen erlangt haben, um ein technisches Interesse zu entwickeln, aber noch nicht genug Wissen, um technische Probleme lösen zu können. Es zeigte sich, dass Schüler höherer Klassenstufen diese Anforderungen am besten erfüllen.

Aus diesem Grund wurden Schüler von Realschulen und Gymnasien ab etwa der siebenten Klassenstufe als geeignete Testpersonen eingestuft. Eine Übersicht über ihre Altersverteilung ist in Abbildung 56 zu finden.

Als weitere Bedingung für die Teilnahme am Test wurde vorausgesetzt, dass die Schüler noch keine Elektronik- und Programmiererfahrung besitzen. Ein allgemeines technisches Interesse sowie grundlegende PC-Kenntnisse waren jedoch erwünscht.

Unter Berücksichtigung dieser Vorgaben wurden sieben Gymnasiasten und acht Realschüler ausgewählt. Allerdings hatten sich keine Schülerinnen für den Test bereit erklärt, so dass dieser nur mit männlichen Schülern durchgeführt wurde. Es wurden sechs Gruppen zusammengestellt. Diese bestanden aus einmal einer Person, zweimal zwei Personen, zweimal drei Personen und einmal vier Personen. Die größeren Gruppen setzten sich aus jüngeren, die kleineren Gruppen eher aus älteren Schülern zusammen.

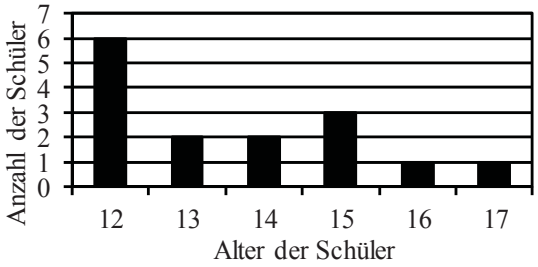

Abbildung 56: Altersverteilung der am Test beteiligten Schüler

5.2.2 Ablauf der Nutzertests

Das Arbeitsumfeld während der Durchführung der Tests sollte so gestaltet sein, dass sie der voraussichtlichen Nutzungssituation im späteren realen Einsatz entspricht. So wurden die Schüler, im Gegensatz zu den in Abschnitt 3.3.3 vorgestellten Tests mit „EasyKit macht Schule“, nicht im Klassenverband betreut, sondern in den einzelnen, vorher festgelegten Gruppen.

Durch das zum Teil geringe Alter der Probanden musste die Testdauer auf maximal eine Stunde beschränkt bleiben, da dies bereits oberhalb der maximalen Konzentrationsdauer dieser Altersgruppe liegt. Da eine Stunde für eine vollständige Untersuchung aller implementierten Bestandteile kurz bemessen ist, mussten verschiedene Prioritäten vergeben und Vereinfachungen getroffen werden. Anhand der Prioritäten wurde die Reihenfolge der zu untersuchenden Bestandteile festgelegt. Es erfolgte eine Aufteilung des Tests in drei Schritte, welche in folgender Reihenfolge zu bearbeiten waren:

- Inbetriebnahme der Hardwareplattform
- Modellierung der Hardware und Software im Ablaufdiagramm und im Datenflussplan
- Erstellung eines neuen Softwarefunktionsbausteins mit Hilfe der dritten Ebene der Programmierung zur Nutzung im Datenflussplan

Während des gesamten Tests wurden die Schüler und ihre Diskussionen beobachtet, wobei sie zusätzlich das Gesehene kommentieren und noch einmal mit eigenen Worten beschreiben sollten. Auf diese Weise kam die in Ab-

schnitt 2.3.5 vorgestellte „Thinking Aloud“-Methode während des gesamten Testverlaufs zum Einsatz.

Abweichend von den Nutzertests mit „EasyKit macht Schule“ wurde hier auch die Unterstützung bei der Entwicklung der Hardware getestet. Hierfür standen den Schülern elektronische Bauteile zur Verfügung, welche miteinander zu verbinden waren, um die gewünschten elektronischen Schaltungen zu realisieren. Aus diesem Grund wurde für den Test ein Baukasten mit dem Mikrocontroller PIC18F2520, unterschiedlichen Widerständen, Kondensatoren, Quarzen, Tastern, Schaltern und lichtemittierenden Dioden zusammengestellt. Dieser ist in Abbildung 57 auf der rechten Seite zu sehen. Die einzelnen Bauteile waren mit ihrem Namen und den jeweiligen Kenngrößen beschriftet. An Widerstände befand sich also beispielsweise die Bezeichnung „Widerstand“ und der entsprechende Widerstandswert. Verschiedene Hersteller bieten vergleichbare Baukästen bereits an. So ist beispielsweise im Conrad Online Shop ein „Profi-Lernpaket Mikrocontroller“ zu finden [Con2013]. Dieses beinhaltet ähnliche Komponenten wie der hier vorgestellte Baukasten.

Genau wie beim Nutzertest mit „EasyKit macht Schule“, erfolgte zunächst die Abfrage einiger statistischer Daten der Schüler wie zum Beispiel das Alter und die Schulbildung. Außerdem sollten die Schüler erklären, ob sie bereits mit der Entwicklung von Hardware oder Software vertraut seien.

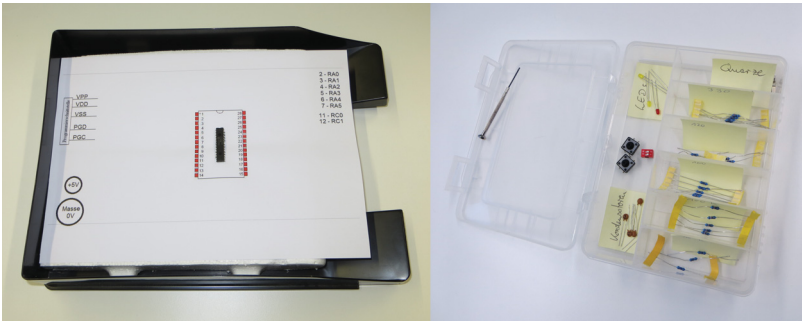

Abbildung 57: Vereinfachtes Steckbrett und Bauteilsortiment zum Aufbau der elektronischen Schaltungen für den Mikrocontroller

Anschließend wurde den Schülern vermittelt, was mechatronische Systeme sind und welche Aufgaben Mikrocontroller innerhalb dieser Systeme übernehmen können. Hierfür wurden einige typische Beispiele des alltäglichen Lebens angesprochen und erklärt.

Inbetriebnahme der Hardwareplattform

Nach der Vermittlung dieser grundlegenden Informationen, begann der eigentliche Test. Die erste Aufgabe für die Schüler bestand darin, die Testsoftware zu starten und die Programmierschaltung für den PIC18F2520 nachzubauen. Hierfür sollten die Schüler den in Abbildung 44 dargestellten Dialog zur Inbetriebnahme der Hardwareplattform nutzen. Um eine solche Schaltung aufbauen zu können, muss ein Programmieradapter zur Verfügung stehen, was im vorliegenden Fall beispielsweise eine ICD3 [Mic2011b] sein kann. Für den Test wurde vorausgesetzt, dass der Nutzer einen solchen Adapter besitzt und dass er die Funktionen der jeweiligen Anschlusskabel der Schnittstelle, zum Beispiel durch Lesen des Handbuchs, in Erfahrung bringen kann.

Aus Sicherheits- und Zeitgründen musste darauf verzichtet werden, die Schaltung mit Hilfe von Lötkolben oder Steckplatinen zu realisieren. Stattdessen erstellten die Schüler die Schaltung auf einem vereinfachten Steckbrett. Dieses bestand aus einer ebenen Fläche aus Polystyrol, in welche die Bauteile hineingesteckt werden konnten. Das Steckbrett ist in Abbildung 57 auf der linken Seite zu sehen.

Die elektrischen Verbindungen wurden durch Linien zwischen den Anschlüssen der Bauteile angedeutet. Der Ablauf des Aufbaus der Schaltungen mit dem vereinfachten Steckbrett ist mit dem Schaltungsaufbau auf einer Lochraster-Platine zu vergleichen. Allerdings war das zeitaufwändige Löten der Verbindungen hier nicht notwendig. Somit konnte die Aufbaudauer für die Schaltungen drastisch reduziert werden. Dies war wichtig, da die Vorgabe der maximalen Testdauer von einer Stunde eingehalten werden musste.

Modellierung von Hard- und Software

Im zweiten Teil sollte ein vollständiges mechatronisches System modelliert werden, so dass aus diesem der Quellcode erzeugt werden kann. Außerdem

sollte parallel hierzu die zugehörige Hardware auf dem Steckbrett aus dem ersten Aufgabenteil realisiert werden.

Hierfür musste den Schülern zunächst eine Einführung in die Grundfunktionen der Testumgebung gegeben werden. Zwar soll die hier vorgestellte Testumgebung die Entwicklung eines mechatronischen Systems erleichtern, jedoch ist es unrealistisch, dass die einzelnen graphischen Programmiersprachen und ihre Zusammenhänge von Nutzern ohne jegliches Expertenwissen intuitiv verstanden werden können. Den Schülern wurde die Funktionsweise des Ablaufdiagramms und der Datenflusspläne erklärt, indem das Beispiel aus Abschnitt 5.1.8 modelliert wurde. Auf die Demonstration des Aufbaus der Hardware auf dem Steckbrett wurde verzichtet.

Eine solche Einführung kann späteren Nutzern in Form eines Videos oder auch integriert in die Entwicklungsoberfläche zur Verfügung gestellt werden. Die wichtigsten Informationen über die zentralen Bestandteile der Software können dabei in wenigen Minuten vermittelt werden.

Nach der kurzen Vorstellung konnten die Nutzer mit der Entwicklung ihres eigenen mechatronischen Systems beginnen. Hierfür erhielten sie die Aufgabe, eine lichtemittierende Diode (LED) abwechselnd für die Dauer von einer Sekunde an- und anschließend genauso lange auszuschalten. Hierfür sollten die Nutzer zunächst zwei Zustände in das Ablaufdiagramm einfügen und parametrieren. Jedem Zustand sollte ein Datenflussplan zugewiesen werden, in welchem anschließend die Hard- und Softwaremodellierung erfolgen sollte. Der Nutzer muss also das Ablaufdiagramm sowie die Hardware-, Software- und Schnittstellenebene der Datenflusspläne manipulieren. Auf diese Weise konnte der Hauptprogrammierbereich der Umgebung getestet werden.

In diesem Teil des Tests wurden die Schüler besonders intensiv dazu aufgefordert, die vorhandenen Systembestandteile noch einmal mit ihren eigenen Worten zu beschreiben. So entstand ein umfassendes Bild darüber, wie die einzelnen Bestandteile der Testumgebung verstanden wurden.

Nach Fertigstellung der Software wurde diese vom Tester auf ihre Funktionsfähigkeit überprüft. Wurde die gewünschte Funktionalität nicht erreicht, so beschrieb der Tester, wie sich das entwickelte System verhalten würde.

Dies war nötig, da in der Testumgebung noch kein Simulationsmodus oder Onlinedebugging, wie in Abschnitt 3.1.2 für EasyKit vorgestellt, implementiert wurde. Es wurde darauf geachtet, dass nur das im Simulationsmodus erkennbare Verhalten wiedergegeben wurde. Auf diese Weise sollte getestet werden, ob die Nutzer die auftretenden Fehler mit Hilfe des Simulationsmodus erkennen und verbessern können.

Erstellung eines neuen Softwarefunktionsbausteins

Auch die Erstellung eines neuen Softwarefunktionsbausteins ist ohne eine kurze Einführung nicht möglich. Daher wurde dies am Beispiel einer Addition von drei Zahlen vorgeführt. Auch dieser Schritt lässt sich sehr gut in Form eines Videos darstellen. In wenigen Minuten sind dabei alle wichtigen Bestandteile der Oberfläche erklärt.

Nach der Einführung in die Oberfläche wurde den Nutzern die Aufgabe gestellt, einen Softwarefunktionsbaustein zu erstellen, welches die Gleichung $25a + b = c$ abbildet. Durch die Multiplikation mit der Zahl 25 mussten die Nutzer eine Möglichkeit finden, eine Konstante zu erzeugen. Die Lösung dieser Problematik wurde genau beobachtet, da sie erste Rückschlüsse zulassen kann, ob die Nutzer mit der Oberfläche arbeiten können.

Da bis zu den Tests noch keine Gültigkeitsprüfung für den Quellcode implementiert wurde, übernahm der Tester diese Funktion. Traten im Quellcode fehlerhafte Programmfragmente auf, so machte er auf diese aufmerksam, indem er kurze Fehlermeldungen ansagte. Diese Meldungen stellten den Testnutzern nur wenige Informationen zur Verfügung, welche sich an den üblichen Debugging-Fehlermeldungen von Compilern orientierten.

In Abbildung 58 ist der Ablauf der Nutzertests noch einmal graphisch dargestellt. Die Abbildung zeigt in Kurzform die zuvor bereitgestellten Informationen, die durchzuführenden Aufgabenstellungen und die wichtigsten während des Tests gestellten Fragen.

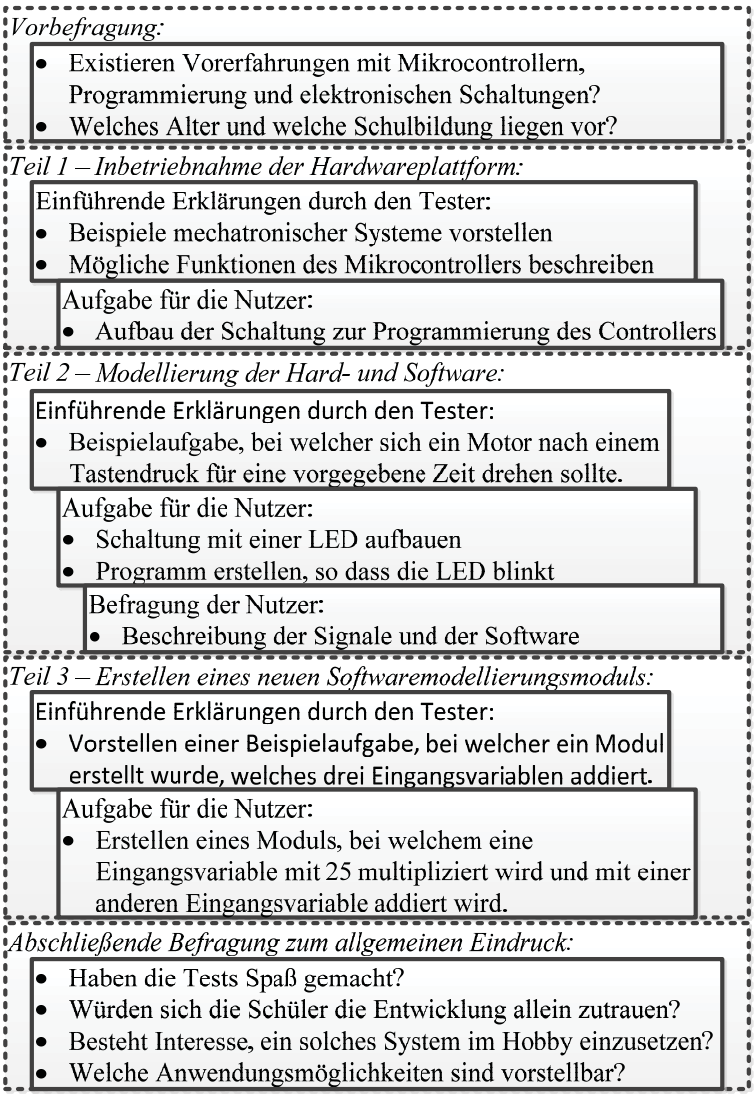

Abbildung 58: Zusammenfassung des Ablaufs der Nutzertests

5.2.3 Ergebnisse der Nutzertests

In der anfänglichen Befragung gaben alle Schüler an, dass sie noch nie ein mikrocontrollerbasiertes mechatronisches System entwickelt hätten.

Bezüglich der Erfahrungen in der Entwicklung elektronischer Schaltungen zeichnete sich ein ähnliches Bild ab. Lediglich ein Schüler erwähnte, dass er eine Lichtorgel aus einem Bausatz zusammengebaut hätte. Lichtorgeln sind Beleuchtungssysteme, welche früher oft in Diskotheken eingesetzt wurden und ihre Helligkeit mit der Amplitude bestimmter Frequenzbereiche veränderten. Der Schüler musste dabei keine Dimensionierung von Bauteilen vornehmen, sondern lediglich die Bauteile nach einem vorgegebenen Schema anordnen, so dass der Begriff Schaltungsentwicklung hier eigentlich nicht zutreffend ist. Dennoch sollte dies hier erwähnt sein, da diese Erfahrung ein besseres Verständnis elektrischer Schaltkreise mit sich bringt.

Die Schüler wurden außerdem gefragt, ob sie bereits in irgendeiner Weise Programmiererfahrungen gesammelt hätten. Ein Schüler gab dabei an, bereits mit LEGO MINDSTORMS NXT gearbeitet zu haben. Ein anderer hatte bereits versucht, eine Software mit Hilfe der Sprache Gambas [Gam2013] zu entwickeln, was er allerdings nach einigen Fehlversuchen aufgab. Ansonsten gaben alle Schüler an, keine Programmiererfahrung zu besitzen.

Inbetriebnahme des Mikrocontrollers

Der Nachbau der Schaltung zur Inbetriebnahme des Mikrocontrollers mit Hilfe des vorgegebenen Schaltplans war für alle Gruppen problemlos möglich. Lediglich eine Gruppe vergaß eine Kabelverbindung zwischen dem Resetschalter und der Masse herzustellen. Der Resetschalter hätte auf diese Weise nicht funktioniert. Die Programmierung des Mikrocontrollers wäre jedoch trotz dieses Fehlers problemlos möglich gewesen.

Die schnellste Gruppe benötigte für den Aufbau weniger als acht Minuten. Der Durchschnitt lag bei etwas mehr als elf Minuten. Die beiden Gruppen mit den jüngsten Schülern benötigten im Schnitt etwa 45% mehr Zeit als die anderen Gruppen.

Aus den Diskussionen zeigte sich, dass die Schüler die im Schaltplan genutzten Schaltzeichen für die Bauelemente zuvor nicht kannten. Die hierfür eingefügte Legende wurde daher von allen Gruppen intensiv genutzt. Jedoch

hatten einige Schüler Probleme, die richtigen Bauteile auszuwählen, obwohl die Bezeichnungen eindeutig waren. Zwar wurden diese von den anderen Teilnehmern aus der Gruppe korrigiert, dennoch kann es an dieser Stelle sinnvoll sein, Fotos der Bauteile in die Legende zu integrieren.

Modellierung der Hard- und Software

Wie bereits beschrieben, sollten die Schüler eine LED abwechselnd für eine Sekunde ein- und für eine Sekunde ausschalten. Um das hierfür benötigte mechatronische System zu modellieren, begannen alle Schüler mit der Erweiterung des Ablaufdiagramms. Es wurde von den Schülern erkannt, dass zwei neue Zustände in das Ablaufdiagramm eingefügt werden mussten und dass jeder Zustand nur einen Ausgangsverbinder benötigte. Auch die Zuweisung der Bedingungen am Ausgang des Zustands und die Zuordnung der Datenflusspläne zu den Zuständen stellte für die Schüler kein Hindernis dar.

Ein erstes Problem trat auf, als die Schüler versuchten, die minimale Ausführungsdauer der Zustände einzustellen. Eigentlich muss hier bei beiden Zuständen eine Sekunde eingestellt werden. Manche Schüler waren allerdings zunächst der Meinung, dass dies nur bei einem Zustand sein müsste, während beim zweiten Zustand keine minimale Ausführungsdauer eingestellt werden müsste. Durch Diskussionen zwischen den Schülern konnten diese Fehler jedoch schnell beseitigt werden, so dass am Ende alle Gruppen zum richtigen Ergebnis kamen. Dieses Problem trat nur in den jüngeren Gruppen auf, so dass die Annahme nahe liegt, dass die jüngeren Schüler größere Probleme hatten, sich in die Funktionsweise des zeitlichen Ablaufs des Ablaufdiagramms hinein zu versetzen.

Um die Hardware im Datenflussplan zu modellieren, musste lediglich eine LED in den Hardwarebereich eingefügt und parametriert werden. Die elektrischen Parameter wurden den Schülern dabei vorgegeben, da diese beispielsweise aus einem Datenblatt entnommen werden müssen. Dieser Schritt stellte für die Nutzer keine Hürde dar, jedoch benötigten einige Gruppen etwas Zeit, um die LED in der Bibliothek der Hardwaremodellierungswerkzeuge zu finden. Die Implementierung einer Schnellsuche kann hier Abhilfe schaffen. Bei den jüngeren Gruppen ergab sich die Frage, ob ein Taster oder Schalter für das Programm nötig sei. Diese Frage konnten die Gruppen jedoch auch durch Diskussionen untereinander schnell klären.

Nachdem die Hardware modelliert wurde, sollten die Schüler diese auch auf dem Steckbrett realisieren. Es wurde von allen Gruppen erkannt, dass eine Anpassungsschaltung benötigt wurde. Auch den Schaltplan dieser Anpassungsschaltung anzeigen zu lassen und zu realisieren, stellte für die Schüler kein Hindernis dar. Sie erkannten dabei selbstständig, an welchen Mikrocontrolleranschluss die LED angeschlossen werden musste und bewiesen somit, dass sie die Beziehung zwischen der Schnittstellenebene innerhalb des Datenflussplans und dem eigentlichen Mikrocontroller verstanden hatten.

Um die LED blinken zu lassen, muss in einem Datenflussplan der Anschluss, mit welchem die LED verbunden ist, auf „true“ gesetzt werden, während in dem anderen Datenflussplan der gleiche Anschluss auf „false“ gesetzt werden muss. Den ersten Teil konnten die Schüler fehlerlos realisieren, indem sie eine Konstante mit dem Anschluss verbanden und den Wert der Konstante auf „true“ setzten. Auch die Auswahl des Datentyps stellte für die Testnutzer keine große Herausforderung dar. Allerdings vergaß eine Gruppe, dass auch der Anschluss im zweiten Datenflussplan verbunden werden musste. Außerdem wurde von zwei anderen Gruppen vergessen, dass den Variablen, welche als Bedingungen an den Ausgangsverbindern der Zustände des Ablaufdiagramms festgelegt wurden, ebenso ein Wert zugewiesen werden musste, um in den an diesen Verbinder angeschlossene Zustand springen zu können.

Beide Fehler stellen ungültige Zustände dar. In der zum Test genutzten Version wurden diese Fehler noch nicht durch die Software erkannt. Die Wahrscheinlichkeit des Auftretens solcher Fehler ist signifikant und als kritisch einzustufen, so dass bei der Weiterentwicklung der Software eine Prüfung der Plausibilität der Datenflusspläne implementiert werden muss. Diese muss überprüfen, ob allen benötigten Variablen und allen angeschlossenen Aktoren entsprechende Werte zugewiesen wurden.

Allerdings konnten die Schüler diese Probleme selbst lösen, nachdem das von ihnen programmierte Verhalten simuliert wurde. Wie bereits beschrieben, wurde die Simulation durch den Tester nachgestellt, indem er den Schülern erklärte, wie sich das System verhalten würde. Mit Hilfe dieser Informationen konnten die Fehler schnell gefunden werden. Bei der Simula-

tion wurde davon ausgegangen, dass den Aktoren kein neuer Wert zugewiesen wird, sondern der zuvor zugewiesene Wert bestehen bleibt. Für die nicht verbundenen Variablenfunktionsbausteine wurde angenommen, dass der jeweilige Standardwert genutzt wird, was beim Datentyp „bool“ dem Wert „false“ entspricht. Diese Annahmen entsprechen dem realistischen Verhalten des mit der Software erzeugten Quellcodes. Auf diese Weise erreichten alle Gruppen das Ziel, das System auf dem Steckbrett aufzubauen und in der Testumgebung zu modellieren und zu programmieren. Auch hier benötigten die jüngeren Gruppen deutlich mehr Zeit.

Abschließend zu diesem Teil des Tests wurden die Schüler darum gebeten, die Beziehung zwischen dem Software- und dem Hardwarebereich am Beispiel der LED zu erklären. Es zeigte sich, dass alle Gruppen die Beziehung zwischen der elektrischen Größe an der Hardware und der Repräsentation dieser Größe als Datentyp im Softwarebereich verstanden hatten.

Erstellung eines neuen Softwarefunktionsbausteins

Wie bereits beschrieben, sollten die Nutzer im dritten Teil des Tests selbstständig einen Softwarefunktionsbaustein erstellen. Die jüngste Gruppe wurde von dieser Aufgabe ausgenommen, da zum einen ihre maximale Aufmerksamkeitsdauer bereits erreicht war und zum anderen die zu testende dritte Ebene der Programmierung in erster Linie für die Nutzung durch fortgeschrittene Anwender vorgesehen ist.

Wie bereits beschrieben, sollte die Gleichung $25a + b = c$ abgebildet werden. Am Einfachsten lässt sich dies in zwei Zeilen durch eine Multiplikation, gefolgt von einer Addition, darstellen. Das Einfügen der beiden hierfür benötigten Funktionsbausteine konnte von allen Gruppen gemeistert werden. Eine Gruppe hatte die Aufgabenstellung zunächst jedoch nicht richtig verstanden. Daher wurde ihnen zusätzlich die in Abbildung 59 dargestellte Skizze gezeigt. Anschließend konnten auch sie die beiden Funktionsbausteine einfügen. Wie erwartet, traten zunächst Probleme beim Einsetzen der Zahl auf. Eine Gruppe entschloss sich jedoch sehr schnell, einen Multiplikator des Multiplikationsschrittes im Quelltext durch die Zahl zu ersetzen, wodurch das Ziel hier bereits erreicht war.

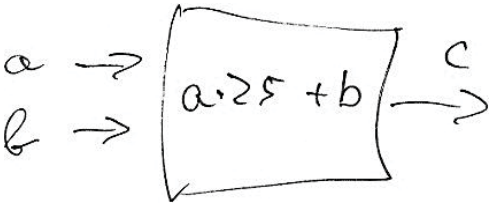

Abbildung 59: Skizze des während des Nutzertests zu entwerfenden Funktionsbausteins

Drei Gruppen durchsuchten die Softwaremodellierungswerkzeuge und fanden den Funktionsbaustein „Konstante“. Diesen fügten sie ein, setzten den Wert der Konstante auf 25 und ersetzten einen Multiplikator durch die Konstante. Lediglich eine Gruppe erreichte das Ziel nicht. Sie gaben hierfür Konzentrationsschwierigkeiten an, weshalb der Versuch mit dieser Gruppe abgebrochen wurde. Abbildung 60 zeigt, welche Altersgruppen der Schüler die Teilabschnitte des Tests erfolgreich absolvierten.

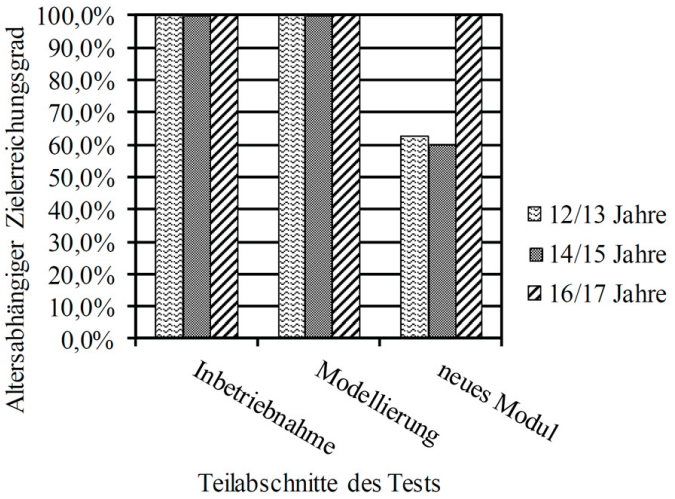

Abbildung 60: Übersicht über den Grad der Zielerreichung in den drei Teilen des Nutzertests

Die vier Gruppen, welche bis zu diesem Punkt gekommen waren, mussten nun nur noch festlegen, welche Variablen als Ein- und Ausgangsverbinder des neuen Funktionsbausteins festgelegt werden sollten. Dies bereitete keine Schwierigkeiten. Auch der Einsatz der Werkzeuge zum Umbenennen und Ersetzen der Variablen stellte die Schüler vor keine großen Herausforderungen.

Abschließende Befragung

Nachdem alle Gruppen den Test beendet hatten, wurden sie nach dem allgemeinen Eindruck zur Software und zum Test befragt. Alle Schüler gaben an, dass ihnen die Tests Spaß gemacht hätten und dass sie sich grundsätzlich vorstellen könnten, ein solches System in der Freizeit zu nutzen. Zwei Schüler merkten allerdings an, dass sie bisher kein Hobby betreiben würden, in welchem ein solcher Einsatz möglich wäre und dass ihr technisches Interesse auch nicht so ausgeprägt sei, um ein Hobby in einem solchen Bereich zu beginnen. Andere Schüler zählten hingegen bereitwillig verschiedene Anwendungsmöglichkeiten aus ihrem näheren Umfeld auf und zeigten ihre Bereitschaft, diese auch zu realisieren. In Abbildung 61 ist die Verteilung des Interesses der Schüler an einer weiteren Nutzung des Entwicklungssystems dargestellt.

Den Schülern wurde klar gemacht, dass fehlerhafte Schaltungen auch zur Zerstörung des Systems führen könnten. Dennoch erklärten sie, dass sie sich in der Lage fühlten, die Entwicklung eines mechatronischen Systems mit Hilfe eines solchen Entwicklungssystems auch ohne Beteiligung einer fachkundigen Person durchzuführen. Die Angst bei der erstmaligen Entwicklung eines nichtmodulbasierten mechatronischen Systems einen Fehler zu machen, wird durch dieses Entwicklungssystem also deutlich reduziert.

Vor allem die jüngeren Schüler wiesen allerdings darauf hin, dass sie hierfür lieber in einer Gruppe zusammen mit anderen Schülern arbeiten würden. Abbildung 62 zeigt eine Übersicht, wie viele Nutzer sich eine individuelle Arbeit mit dem System vorstellen könnten und wie viele die Arbeit in einer Gruppe bevorzugen würden. Auch wenn viele Schüler die Gruppenarbeit bevorzugen, zeigt sich doch, dass die Einstiegsbarrieren in das Themengebiet der Mechatronik mit Hilfe des hier vorgestellten Systems gesenkt werden können.

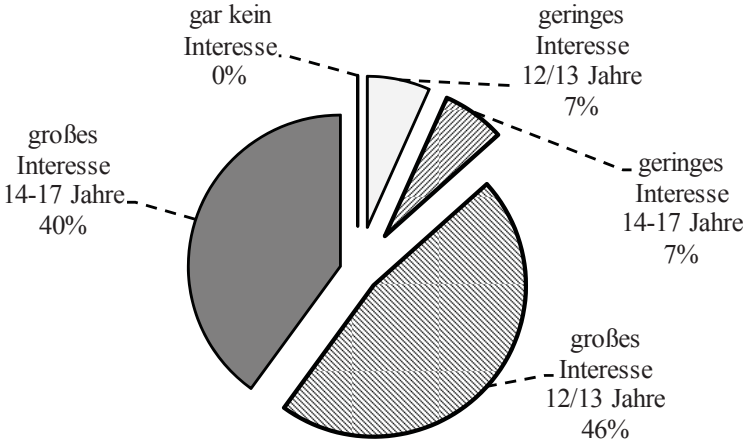

Abbildung 61: Verteilung des Interesses der Schüler an der weiteren Arbeit mit mikrocontrollerbasierten mechatronischen Systemen

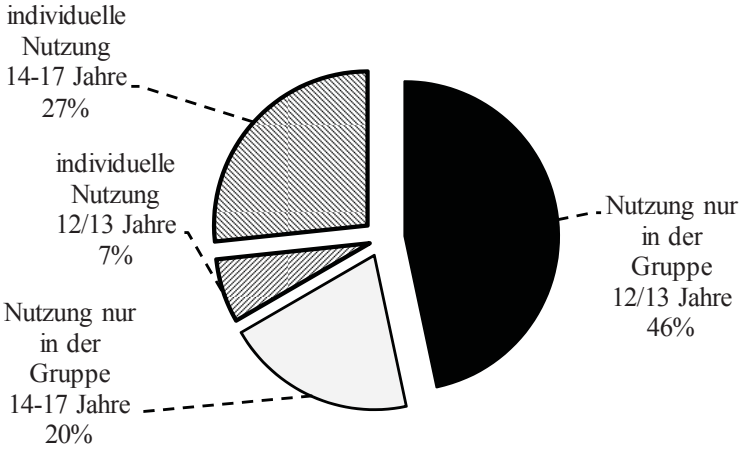

Abbildung 62: Übersicht über die Verteilung der durch die Schüler angestrebten Nutzungsart nach individueller oder Gruppennutzung

5.3 Zusammenfassung

In diesem Kapitel wurde zunächst vorgestellt, in welcher Weise die Umsetzungskonzepte aus dem vorigen Kapitel in die Testentwicklungsumgebung EasyHobby implementiert wurden. Dabei standen die Realisierung der Modellierung der Hardware und die dritte Ebene der Softwareentwicklung im Zentrum. Aber auch die sonstigen Erweiterungen, welche die Nutzbarkeit des Systems steigern sollten, wurden hier vorgestellt.

Im zweiten Teil standen die Nutzertests im Zentrum. Zunächst wurden hierfür die Auswahl der Nutzer und der Aufbau des Tests beschrieben. Anschließend wurden die Ergebnisse der Tests vorgestellt. Wichtige Aussagen sind:

- Der Aufbau der Hardwarebestandteile bereitete keine Schwierigkeiten.
- Auch die Erweiterung des Ablaufdiagramms war für die Schüler möglich.
- Die Modellierung der Hardware und der Software im Datenflussplan konnte ebenso von allen Schülern realisiert werden. Die einzigen hierbei aufgetretenen Probleme können in zukünftigen Versionen durch eine einfache Prüfung der Plausibilität der Datenflusspläne abgefangen werden.
- Die Nutzung der dritten Ebene der Programmierung stellte die größte Herausforderung dar. Dennoch waren 80% der Testgruppen hierbei erfolgreich.
- Grundsätzlich kann gesagt werden, dass Schüler aller getesteten Altersklassen fähig sind, die Hauptbestandteile eines solchen Entwicklungssystems erfolgreich zu nutzen. Dabei hängt der Erfolg jedoch stark vom Interesse, der Konzentrationsfähigkeit und dem logischen Denkvermögen der Nutzer ab.

- Die jüngeren Schüler hatten mehr Probleme und benötigten mehr Zeit bei der Arbeit mit dem System als die älteren Schüler. Es ist daher sinnvoll, wenn jüngere Nutzer bis etwa 15 Jahre in kleinen Gruppen oder unter Aufsicht arbeiten, da einige Verständnisprobleme durch Diskussionen untereinander gelöst werden können.
- Die Motivation eines Teils der Schüler, sich auch außerhalb der Schule mit diesem Themengebiet zu beschäftigen, ist ein wichtiger Fortschritt. Dies resultiert vor allem aus der umfassenden Unterstützung bei der Entwicklung der Hard- und der Software des mechatronischen Systems, wodurch die Angst vor Fehlern während der Entwicklung reduziert wird. Wie bei vielen Einstiegssystemen gilt auch hier, dass die Motivation des Nutzers von zentraler Bedeutung ist.

In Abbildung 20 und in Abbildung 21 wurden Zielstellungen dargestellt, nach welchen eine hohe Flexibilität der Hard- und Softwareentwicklung für Nutzer mit wenig oder gar keinem domänenspezifischen Expertenwissen ermöglicht werden sollte. Natürlich konnte die Flexibilität von professionellen textbasierten Umgebungen zur Softwareentwicklung nicht erreicht werden, da einfach nutzbare graphische Methoden immer auf Abstraktionen zurückgreifen müssen und somit nicht jede erdenklich Funktion zur Verfügung stellen können. Allerdings geht die realisierbare Funktionalität eines mit dem Drei-Ebenen-Konzept modellierten Mikrocontrollerprogramms deutlich über die mögliche Funktionalität eines durch eine einschichtige Lösung erzeugten Programms hinaus.

Auch die Hardwareentwicklung ist natürlich nicht so flexibel, wie die grundlegende Entwicklung einer Schaltung durch einen Elektronikspezialisten. Aber auch hier war eine deutliche Steigerung im Vergleich zu rein modulbasierten Ansätzen möglich. Die einfache Anwendbarkeit durch die Zielgruppe wurde in den Nutzertests unter Beweis gestellt.

Damit kann also gesagt werden, dass die gesteckten Ziele erreicht wurden. Weiterhin zeigten die Nutzertests, dass die Modellierungsmöglichkeiten der Hardware innerhalb der Software tatsächlich das technische Verständnis, insbesondere der Signalarten und der Signalwandlung am Mikrocontroller, fördern können.

6 Zusammenfassung und Ausblick

Im Mittelpunkt dieser Arbeit standen Untersuchungen über mögliche Maßnahmen zur Gestaltung von Entwicklungssystemen, welche Nutzer bei einer flexiblen und weitestgehend plattformunabhängigen Entwicklung mikrocontrollerbasierter mechatronischer Systeme unterstützen. Zur Entwicklung solcher Systeme gehören dabei die Realisierung der benötigten elektronischen Schaltungen und die Programmierung des darin enthaltenen Mikrocontrollers. Beide Schritte sollen möglichst mikrocontrollernah erfolgen, um eine hohe Flexibilität zu gewährleisten und um den Nutzern einen Einblick in die Funktionsweise der Mikrocontrollertechnik zu ermöglichen. Ziel war es außerdem, dass die Nutzer hierfür über kein oder nur sehr grundlegendes domänenspezifisches Expertenwissen auf den Gebieten der Elektronik und Informationstechnik verfügen müssen.

6.1 Zusammenfassung

Der erste Teil der Arbeit widmet sich den wichtigsten Grundlagen mikrocontrollerbasierter mechatronischer Systeme. Hierfür wurde zunächst der Begriff der mechatronischen Systeme sowie deren Aufbau und Entwurf beschrieben. Anschließend erfolgte eine Erweiterung dieses Begriffs auf die mikrocontrollerbasierten Systeme. In diesem Zusammenhang wurde der Begriff des domänenspezifischen Expertenwissens im Bereich der Mechatronik erklärt. Dem schloss sich eine Vermittlung grundlegender Informationen zur Programmierung von Mikrocontrollern an, wobei vor allem die Begriffe der Plattformunabhängigkeit, der modellbasierten Softwareentwicklung und der graphischen Programmierung im Zentrum des Diskussion standen. Weiterhin wurde ein Überblick über das Themengebiet der Mensch-Maschine-Schnittstelle gegeben. Dies beinhaltete eine Beschreibung der in dieser Arbeit eingesetzten Methoden der Usability-Evaluierung. Es folgte eine Unter-

suchung bereits existierender Entwicklungssysteme für mikrocontrollerbasierte mechatronische Systeme, wobei an dieser Stelle reine Programmierumgebungen genauso im Fokus standen wie Systeme, welche den gesamten Entwicklungsprozess mikrocontrollerbasierter mechatronischer Systeme unterstützen. Auf diese Weise konnten Defizite der existierenden Systeme erkannt werden, was die wichtigste Grundlage dieser Arbeit darstellt.

Auf Basis der zusammengetragenen Defizite wurden im Rahmen des Projekts EasyKit die ersten Lösungsansätze für eine Vereinfachung der Programmierung von mikrocontrollerbasierten Systemen auf der Mikrocontrollerebene entwickelt. Nach der Vorstellung der Hard- und Softwarebestandteile der industriellen Version von EasyKit wurden die Komponenten des EasyKit Starters vorgestellt. Einen wichtigen Bestandteil des Gesamtsystems stellt dabei ein web-based Training dar, durch welches wichtiges Grundlagenwissen bereitgestellt wird. Mit diesem System, allerdings ohne das web-based Training, konnten erste Nutzertests mit unterschiedlichen Nutzergruppen durchgeführt werden. Dabei konnte nachgewiesen werden, dass der EasyKit Starter grundsätzlich auch durch Anwender ohne ingenieurswissenschaftliches Expertenwissen eingesetzt werden kann. Erstmals konnte diesen Nutzern eine einfache Möglichkeit zur Programmierung auf der Mikrocontrollerebene zur Verfügung gestellt werden. Die Ergebnisse dieser Tests zeigten dennoch weitere Defizite auf, welche in erster Linie auf der Seite der Hardwareentwicklung angesiedelt waren.

Es mussten neue Ansätze zur Beseitigung dieser Defizite erarbeitet werden. Dazu wurden Anforderungen formuliert und neue Realisierungskonzepte entwickelt. Ein Realisierungskonzept beinhaltete erstmalig die Integration einer Repräsentation der elektronischen Hardware auf Mikrocontroller- und Schaltungsebene innerhalb der Entwicklungsumgebung. Somit kann eine integrierte Hard- und Softwaremodellierung innerhalb einer einzigen Entwicklungsumgebung erfolgen, wodurch die Eigenschaften der genutzten Plattform sowie der Sensoren und Aktoren in die Softwaremodellierung einfließen können. Eine besondere Rolle kommt an dieser Stelle der Schnittstellenebene zwischen Hard- und Softwarebereich zu, welche die Wandlung zwischen den elektrischen Signalen des Hardwarebereichs und der entsprechenden Repräsentation in Form von Datentypen des Softwarebereichs

übernimmt. Um die Hardwaremodellierung zu ermöglichen, wurde eine Vielzahl in dieser Arbeit vorgestellter Applikationen realisiert.

Ein weiteres vorgestelltes Realisierungskonzept sieht die Nutzung einer hybriden Programmierumgebung vor, welche graphische Methoden zur Programmierung anbietet, jedoch eine editierbare textuelle Darstellung des Quellcodes besitzt. Auf diese Weise kann eine hohe Flexibilität bei gleichzeitiger guter Nutzerfreundlichkeit erreicht werden. Zudem ermöglicht diese Kombination erstmals das einfache Erlernen der Syntax einer textuellen Programmiersprache, indem graphische Elemente eingesetzt werden.

Weitere vorgestellte Realisierungskonzepte beinhalten Ergebnisse von Untersuchungen darüber, welches Wissen Nutzern ohne domänenspezifisches Expertenwissen zur Verfügung gestellt werden muss und wie dieses aufbereitet werden kann.

Die Realisierungskonzepte stellen in ihrer Gesamtheit erstmals ein umfassendes Gesamtkonzept dar, welches Nutzer ohne domänenspezifisches Expertenwissen innerhalb einer einzigen Entwicklungsumgebung beim Einstieg in die Entwicklung mikrocontrollerbasierter mechatronischer Systeme unterstützt. Das Gesamtkonzept reicht dabei von der Inbetriebnahme der informationstechnischen Plattform über die Anbindung von Sensoren und Aktoren bis hin zur eigentlichen Programmierung mit Hilfe graphischer Modellierungssprachen. Dabei steht in jedem Schritt erstmals das Verstehen der jeweiligen auftretenden Probleme im Zentrum und nicht das Ausblenden schwieriger Sachverhalte durch eine Modularisierung. Auf diese Weise kann die Lücke zwischen einfach nutzbaren modulbasierten Entwicklungssystemen und flexiblen modellbasierten Expertensystemen geschlossen werden. Zudem ist trotz der einfachen Einsetzbarkeit erstmals eine weitestgehend plattformunabhängige Entwicklung möglich.

Im Anschluss an die Vorstellung der Konzepte konnten die erfolgversprechendsten Ansätze in eine Testumgebung implementiert werden. Dabei erwies es sich als sinnvoll, viele Bestandteile der Entwicklungsumgebung EasyLab aufzugreifen und durch neue Komponenten zu erweitern oder zu ersetzen.

Mit der so realisierten Testentwicklungsumgebung erfolgten Nutzertests. Da als wichtigste Nutzergruppe vor allem junge, technikbegeisterte Personen angesprochen werden sollen, bestand die Kreis der Testnutzer aus Schülern der mittleren bis oberen Klassenstufen. Die Ergebnisse der Tests zeigten, dass die Schüler durch die Implementierung der Ansätze fähig waren, die Hardware und die Software für mikrocontrollerbasierte mechatronische Systeme auf der Mikrocontrollerebene zu entwickeln. Es konnte gezeigt werden, dass die Integration der Hardwaredarstellung in die Entwicklungsumgebung das Verständnis der Eigenschaften spezieller Signalarten und der Funktionalität der Peripherie des Mikrocontrollers ermöglichte. Außerdem konnte den Testnutzern die Angst vor dem Einarbeitungsaufwand in die Themen der Mikrocontrollertechnik und der Mechatronik genommen werden, da die Entwicklung des Systems für die Schüler einfach war und sie in jedem Schritt auf mögliche Probleme hingewiesen wurden. Ein weiterer positiver Effekt ist, dass die Nutzer sich selbstständig Gedanken über weitere Anwendungen in unterschiedlichen Bereichen machten, was zeigt, dass sie die flexiblen Einsatzmöglichkeiten von Mikrocontrollern weitestgehend verstanden hatten.

Das Ziel, auch Nutzern ohne domänenspezifisches Expertenwissen den Einstieg in die flexible Entwicklung der mikrocontrollerbasierten mechatronischen Systeme auf der Mikrocontroller- und Schaltungsebene zu ermöglichen, wurde daher erreicht.

6.2 Ausblick

Es ist wünschenswert, die untersuchten, implementierten und getesteten Ansätze weiterzuentwickeln und zu nutzen. Eine Möglichkeit ist natürlich die Weiterentwicklung der hier vorgestellten Testumgebung zu einer stabilen und umfassend funktionsfähigen Software. Im Rahmen der Arbeit war eine solche Entwicklung jedoch nicht das Ziel, da der Schwerpunkt der Untersuchungen auf der Entwicklung von Maßnahmen zur Verbesserung der Nutzbarkeit von Entwicklungssystemen für mikrocontrollerbasierte mechatronische Systeme lag. Die folgenden Erweiterungen sind vorstellbar und können die Nutzerfreundlichkeit weiter verbessern.

Mit der im fünften Kapitel vorgestellten Testentwicklungsumgebung kann Quellcode in der Programmiersprache C erzeugt werden. Um diese Entwicklungsumgebung weiterzuentwickeln, kann sie um verschiedene und in anderen Systemen bereits genutzte Bestandteile erweitert werden. So bringt die Integration eines Simulationsmodus, in welchem der Nutzer die Funktion der erstellten Mikrocontrollersoftware in der Entwicklungsumgebung testen kann, viele Vorteile mit sich. Außerdem bietet sich die Implementierung der jeweiligen Toolchains an, um den Quellcode für die jeweiligen Plattformen übersetzen und auf diese laden zu können. Ist dies realisiert, kann eine Möglichkeit vorgesehen werden, mit welcher Daten von der jeweiligen Plattform an die Entwicklungsumgebung gesendet werden, in welcher sie dargestellt werden können, um so ein Onlinedebugging zu ermöglichen.

Im Sinne einer weiteren Verbesserung der Nutzerfreundlichkeit können Vorlagen zum Erstellen des Ablaufdiagramms eingeführt werden. Diese sollten Zustände mit vorgefertigten und vollständig funktionsfähigen Datenflussplänen beinhalten. Darin können Abläufe vorgesehen werden, welche häufig genutzt werden, wie zum Beispiel die Betätigung eines Tasters oder das Einschalten eines Motors. Um bereits beim Einfügen des Zustands in das Ablaufdiagramm auf dessen Funktionsweise hinzuweisen, können verschiedene Parameter, wie die Ausführungszeit oder die Bedingungen am Ausgang des Zustands, vom Nutzer dialogbasiert vorgegeben werden. Nach dem Einfügen können der aus der Vorlage erzeugte Zustand und dessen zugehöriger Datenflussplan bearbeitet werden, so dass auch hier eine hohe Flexibilität gewährleistet ist.

In Abschnitt 4.3.3 wurden verschiedene Systeme vorgestellt, deren Anwendungen als Vorlage für die Modellierungswerkzeuge des Datenflussplans dienten. Die Systeme wurden gemeinsam mit Nutzern aus dem Hobbybereich ausgewählt, so dass die wichtigsten Funktionen zur Verfügung stehen. Es werden jedoch immer wieder neue Applikationen auftreten, welche mit den aktuellen Werkzeugen nicht realisiert werden können. Die Erweiterung der Modellierungswerkzeuge ist daher ein wichtiger Schritt, um neue Sensoren, Aktoren oder Softwarefunktionen einzubinden. Hierfür sind beispielsweise internetbasierte Austauschmöglichkeiten zwischen den Nutzern zu untersuchen.

Auch die dritte Ebene der Programmierung kann durch weitere Werkzeuge verbessert werden, so dass die Anbindung neuer Sensoren und Aktoren einfacher möglich wird. So sind beispielsweise die Möglichkeiten zur Implementierung einer graphischen Oberfläche zur Anbindung von seriellen Schnittstellen (zum Beispiel SPI) zu untersuchen. Viele Analog-Digital-Wandler und kostengünstige Sensoren arbeiten mit solchen Schnittstellen, welche sich oft nur in wenigen Merkmalen unterscheiden. Mit Hilfe einer solchen unterstützenden Oberfläche kann der Nutzer einen Funktionsbaustein erstellen, welcher die Daten der Schnittstelle auslesen kann. Diesen Funktionsbaustein kann er anschließend anderen Nutzern zur Verfügung stellen.

Die Unterstützung der Ansätze zur integrierten Hardware und Software-Entwicklung können ebenso weiter vorangetrieben werden. So ist es beispielsweise vorstellbar, dem Nutzer Bauteilelisten für eine mögliche Bestellung über einen Elektronikversandhandel zur Verfügung zu stellen. Auch kann das Erzeugen von Schaltplänen für Leiterplattenentwurfsprogramme wie zum Beispiel EAGLE [Cad2013] unterstützt werden.

Im praktischen Einsatz werden sich natürlich noch viele weitere Verbesserungsmöglichkeiten ergeben. Dabei muss bei jeder Erweiterung zunächst durchdacht werden, ob diese tatsächlich eine Verbesserung der Nutzerfreundlichkeit mit sich bringt oder ob sie das System für den unerfahrenen Nutzer verkompliziert.

Es bleibt allerdings zu berücksichtigen, dass die Anzahl der Personen aus der möglichen Nutzergruppe, welche bereit sind Geld für ein solches Produkt auszugeben, als vergleichsweise gering angenommen werden muss. Da ein finanzieller Erfolg eines solchen Produkts somit eher fraglich ist, sollte auch die Möglichkeit einer Nutzung der Ansätze in einem anderen Kontext in Erwägung gezogen werden. In einem solchen alternativen Nutzungskontext können sich entscheidende finanzielle Vorteile ergeben, welche den Entwicklungsaufwand rechtfertigen.

Eine Möglichkeit ist die Integration der neuen Ansätze in existierende graphische Entwicklungsumgebungen. Am plausibelsten erscheint die Implementierung in EasyLab. Dafür sind verschiedene Anpassungen notwendig. Den größten Aufwand würde dabei die Umstellung des Simulationsmodus

verursachen, da dieser in der aktuellen Implementierungsvariante nicht mit Funktionsbausteinen arbeitet, welche vom Nutzer selbst definiert wurden.

Die Implementierung in andere graphische und rein ablauforientierte Entwicklungsumgebungen ist zwar vorstellbar, bringt in den meisten Fällen jedoch keine Vorteile mit sich. Meist ist hier bereits die Hardwareentwicklung berücksichtigt, indem sie dem Nutzer in modularer Form zur Verfügung gestellt wird. Vorstellbar wäre jedoch eine Zusatzumgebung, mit welcher für diese Systeme neue Elektronikmodule mit der zugehörigen Software entwickelt werden können.

Am vielversprechendsten erscheint jedoch die weitere Verwendung als zusätzliche Abstraktionsschicht für bereits existierende textuelle Programmierumgebungen, wie sie beispielhaft unter Abschnitt 2.4.1 vorgestellt wurden. Die bereits vorhandene Infrastruktur vereinfacht die Integration dieser Schicht. So sind beispielsweise die Toolchains gewöhnlich bereits vorbereitet. Auch stehen in den meisten Fällen umfassende Informationen über den Mikrocontroller zur Verfügung, so dass das Modell der Schnittstellenebene bereits auf eine umfassende Datenbasis zurückgreifen kann. Eine solche Form der Integration bringt auch Vorteile für die Entwickler der Mikrocontroller mit sich, da sie auf diese Weise auch Nutzern ohne Expertenwissen den Zugang zu ihren Produkten ermöglichen. Somit können Kunden früher und einfacher gebunden werden.

Literaturverzeichnis

[Ard2012a] Arduino: Arduino - ArduinoBoardUno. URL: http://arduino.cc/en/Main/ArduinoBoardUno. - Stand: 29.11.2012.

[Ard2012b] Arduino: Arduino - Software. URL: http://arduino.cc/en/Main/Software. - Stand: 29.11.2012.

[Arm2006] ARM Holdings: An Introduction to the ARM Cortex-M3 Processor. URL: http://www.arm.com/files/pdf/IntroToCortex-M3.pdf. - Stand: 21.11.2012.

[Atm2012] Atmel: Atmel® Studio6 - Two Architectures, One Studio - Overview. URL: http://www.atmel.com/microsite/atmel_studio6/. - Stand: 14.11.2012.

[BA2010a] Bönicke, Holger ; Ament, Christoph: Use of the Mechatronics Dervelopement System Easykit for Didactical Purposes. In: 55th IWK – Internationales Wissenschaftliches Kolloquium : Crossing Borders within the ABC – Automation, Biomedical Engineering and Computer Science, Ilmenau University of Technology, Ilmenau, Germany, 13.-17.09.2010, Verlag ISLE, Ilmenau. ISBN 978-3-938843-53-6. S. 446-451.

[BA2010b] Bönicke, Holger ; Ament, Christoph: Development of an Automation Library for the Mechatronics Development System EasyKit and a Typical Test Application. In: International Conference on Control, Automation and Systems, KINTEX, Gyeonggi-do, Korea, 27.-30.10.2010, IEEE Catalog Number: CFP1010D-CDR. ISBN 978-89-93215-02-1 98560/10/$15 ©ICROS, ISSN: 2093-7121. S. 1352-1356.

[BA2010c] Bönicke, Holger ; Ament, Christoph: Application of the “EasyKit” development method for control of mechatronical systems of small and medium complexity. In: 2010 IEEE International Conference on Control Applications (CCA), Yokohama, Japan, 08.-10.09.2010. ISBN 978-1-4244-5362-7, ISSN: 1085-1992. S. 316-321.

[BA2012] Bönicke, Holger ; Ament, Christoph: Increased Usability through User Interface Modification of Development and Education Tools for Embedded Systems In: 2012 IEEE International Conference on Control Applications (CCA), Dubrovnik, Croatia, 03.-05.10.2012. ISBN 978-1-4673-4503-3, ISSN: 1085-1992. S. 675-680.

[BG2008] Bröckl, Ulrich ; Goll, Joachim: C als erste Programmiersprache : vom Einsteiger zum Profi. 6., überarbeitete Auflage, Wiesbaden: Vieweg+Teubner Verlag, 2008. ISBN 3-8351-0222-2, 978-3-8351-0222-4.

[BGB2008] Barner, Simon ; Geisinger, Michael ; Buckl, Christian ; Knoll, Alois: EasyLab : Model-Based Development of Software for Mechatronic Systems. In: IEEE/ASAME International Conference on Mechatronic and Embedded Systems and Applications, Beijing, China, 12.-15.10.2008.

[BGH2010] Barner, Simon ; Geisinger, Michael ; Huang, Jia ; Knoll, Alois ; Bönicke, Holger ; Ament, Christoph ; Mades, Jochen ; Pittschellis, Rheinhard ; Bauer, Gerd: EasyKit : Eine allgemeine Methodik für die Entwicklung von Steuerungskomponenten. In: Gausemeier, Jürgen ; Ramming, Franz ; Schäfer, Wilhelm ; Trächtler, Ansgar (Hrsg): Entwurf mechatronischer SystemeVolume 272, Paderborn, Germany, 18.-19.03.2010, HNI-Verlagsschriftenreihe, Paderborn. ISBN 978-3-18-092099-3. S. 23-36.

[BH2011] Beierlein, Thomas ; Hagenbruch, Olaf: Taschenbuch Mikroprozessortechnik. München: Fachbuchverlag Leipzig im Carl-Hanser-Verlag, 2011. ISBN 978-3-446-42331-2.

[BHK2009] Bollow, Friedrich ; Homann, Matthias ; Köhn, Klaus-Peter: C und C++ für Embedded Systems : [Hardwareübersicht für die Mikrocontroller HC08, C166/C167 und ATMEL ATmega ; Einführung in ARM Cortex-M3 ; großer Praxisteil mit zahlreichen Aufgaben und Lösungen]. 3., aktualisierte und erw. Aufl., Heidelberg: mitp, REDLINE, 2009. ISBN 978-3-8266-5949-2.

[Bia2011] Bias, Randolph G.: A Curriculum for Usability Professionals (and Why We Need One). In: Khalid, Halimahtun ; Hedge, Alan ; Ahram, Tareq Z. (Hrsg.): Advances in Ergonomics Modeling and Usability Evaluation. Boca Raton [u.a.]: CRC Press, Taylor & Francis Group, 2011. ISBN 978-1-439-83503-6. S. 639-646.

[BM2005] Bias, Randolph G. ; Mayhew, Deborah J.: Cost-Justifying Usability : An Update for the Internet Age. Amsterdam [u.a.]: Elsevier [u.a.], 2005. ISBN 0-12-095811-2.

[BPK2009] Bauer, Gerd ; Pittschellis, Rheinhard ; Knoll, Alois: EasyKit : Entwicklungsmethodik für den Entwurf mikromechatronischer Systeme. In: VDE (Hrsg): MikroSystemTechnik - Kongress 2009Volume 3, Berlin, Germany, 12.-14.10.2009, VDE Verlag GmbH, Berlin-Offenbach. ISBN 978-3-8007-3183-1.

[BS2010] Berns, Karsten ; Schmidt, Daniel: Programmierung mit LEGO Mindstorms NXT : Robotersysteme, Entwurfsmethodik, Algorithmen. Berlin; Heidelberg: Springer Verlag, 2010. ISBN 978-3-642-05469-3.

[BST2010] Berns, Karsten ; Schürmann, Bernd ; Trapp, Mario: Eingebettete Systeme : Systemgrundlagen und Entwicklung eingebetteter Software. 1. Auflage, Wiesbaden: Vieweg+Teubner Verlag | Springer Fachmedien, 2010. ISBN 978-3-8348-0422-8.

[Cad2013] CadSoft Computer GmbH: EAGLE PCB Software - Schaltplan, Layout Editor und Autorouter. URL: http://www.cadsoft.de/eagle-pcb-design-software/product-overview/?language=de. - Stand: 16.02.2013.

[Che2009] CHEVAL - Chur Evaluation Laboratory: Methodenkategorien. URL: http://www.cheval-lab.ch/cheval-wissensbasis/methodenkategorien/. - Stand: 07.12.2011.

[Cle1999] Clein, Dan: CMOS IC Layout : concepts, methodologies and tools. Boston [u.a.]: Newnes, 1999. ISBN 0-7506-7194-7.

[Con2010] Conrad Electronic SE: C-Control - C-Control, was sonst? URL: http://www.c-control.de/c-control,_was_sonst.html. - Stand: 19.11.2012.

[Con2013] Conrad Electronic: Conrad Profi-Lernpaket Mikrocontroller im Conrad Online Shop | 192286. URL: http://www.conrad.de/ce/de/product/192286/Conrad-Profi-Lernpaket-Mikrocontroller. - Stand: 17.01.2013.

[Coy2008] Coy, Wolfgang: Auf dem Weg zum "finalen Interface" : Ein medienhistorischer Essay. In: Hellige, Hans D. (Hrsg.): Mensch-Computer-Interface - Zur Geschichte und Zukunft der Computerbedienung. Bielefeld: transcript-Verlag, 2008. ISBN 3-89942-564-2. S. 309-321.

[DF2009] Dumas, Joseph S. ; Fox, Jean E.: Usability Testing : Current Practice and Future Directions. In: Sears, Andrew ; Jacko, Julie A. (Hrsg.): Human-Computer Interaction : Development Process. Boca Raton, Fla. [u.a.]: CRC Press, Taylor & Francis Group, 2009. ISBN 978-1-4200-8890-8. S. 231-251.

[Dig2013] Digia: Qt. URL: http://qt.digia.com/. - Stand: 02.01.2013.

[DIN1946-6-2009] Norm, Deutsches Institut für Normung e. V.: DIN 1946-6. Raumlufttechnik : Teil 6: Lüftung von Wohnungen - Allgemeine Anforderungen, Anforderungen zur Bemessung, Ausführung und Kennzeichnung, Übergabe/Übernahme (Abnahme) und Instandhaltung, Beuth Verlag GmbH, 2009.

[DIN9241-11-1999] Norm, Deutsches Institut für Normung e. V.: DIN EN ISO 9241-11. Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten : Teil 11: Anforderungen an die Gebrauchstauglichkeit; Leitsätze (ISO 9241-11:1998); Deutsche Fassung EN ISO 9241-11:1998, Beuth Verlag GmbH, 1999.

[DIN9241-100-2010] Norm, Deutsches Institut für Normung e. V.: DIN ISO/TR 9241-100. Ergonomie der Mensch-System-Interaktion : Teil 100: Überblick über Normen zur Software-Ergonomie (ISO/TR 9241-100:2010); Deutsche Fassung EN ISO/TR 9241-100:2010, Beuth Verlag GmbH, 2010.

[DIN9241-110-2008] Norm, Deutsches Institut für Normung e. V.: DIN EN ISO 9241-110. Ergonomie der Mensch-System-Interaktion : Teil 110: Grundsätze der Dialoggestaltung (ISO 9241-110:2006); Deutsche Fassung EN ISO 9241-110:2006, Beuth Verlag GmbH, 2008.

[DIN61131-3-2003] Norm, Deutsches Institut für Normung e. V.: DIN EN 61131. Speicherprogrammierbare Steuerungen : Teil 3: Programmiersprachen Beuth Verlag GmbH, 2003.

[DIN66001-1983] Norm, Deutsches Institut für Normung e. V.: DIN 66001. Informationsverarbeitung; Sinnbilder und ihre Anwendung, Beuth Verlag GmbH, 1983-12-00.

[DIN66261-1985] Norm, Deutsches Institut für Normung e. V.: DIN 66261. Informationsverarbeitung ; Sinnbilder für Struktogramme nach Nassi-Shneiderman, Beuth Verlag GmbH, 1985.

[DIN/IEC60381-1-1985] Norm, Deutsches Institut für Normung e. V.: DIN IEC 60381-1. Analoge Signale für Regel- und Steueranlagen; Analoge Gleichstromsignale; Identisch mit IEC 60381-1, Ausgabe 1982, Beuth Verlag GmbH, 1985.

[DIN/IEC60381-2-1980] Norm, Deutsches Institut für Normung e. V.: DIN IEC 60381-2. Analoge Signale für Regel- und Steueranlagen; Teil 2: Analoge Gleichsspannungssignale, Beuth Verlag GmbH, 1980.

[Dör1992] Dörner, Dietrich: Die Logik des Mißlingens : Strategisches Denken in komplexen Situationen. Reinebek bei Hamburg: Rowohlt Taschenbuch Verlag GmbH, 1992. ISBN 3-499-19314-0.

[Dub2009] Dubey, Rahul: Introduction to embedded system design using field programmable gate arrays. London: Springer Verlag, 2009. ISBN 978-1-8488-2015-9.

[Eas2011] EasyKit: Der schnelle Weg zu modularen Mikrosystemen. URL: http://www.easy-kit.de/EasyKit. - Stand: 16.11.2011.

[Fes2010] Festo Didactic: Gerätesatz TP 1021 : EasyKit Mikrocontroller - Grundlagen Elektrotechnik/Elektronik - Elektrotechnik/Elektronik/SPS - Lernsysteme - Festo Didactic. URL: http://www.festo-didactic.com/de-de/lernsysteme/elektrotechnik-elektronik-sps/grundlagen-elektrotechnik-elektronik/geraetesatz-tp-1021-easykit-mikrocontroller.htm?fbid=ZGUuZGUuNTQ0LjEzLjE4LjEyNDMuNzM0Ng. - Stand: 23.02.2012.

[Fes2013] Festo Didactic GmbH & Co. KG: Projektbaukästen EduKit PA - Prozessautomation, Regelungstechnik - Lernsysteme - Festo Didactic. URL: http://www.festo-didactic.com/de-de/lernsysteme/prozessautomation,regelungstechnik/projektbaukaesten-edukit-pa/. - Stand: 02.01.2013.

[Fra2007] Fraunhofer Institut : Intelligente Analyse- und Informationssysteme (IAIS): Roberta : Abschlussbericht. URL: http://www.iais.fraunhofer.de/uploads/media/Abschlussbericht_Roberta_2007-11-21.pdf. - Stand: 13.01.2012.

[Fra2009] Fraunhofer Institut : Intelligente Analyse- und Informationssysteme (IAIS): Roberta : Lernen mit Robotern. URL: http://www.roberta-home.de/de. - Stand: 13.01.2012.

[Fra2010] Fraunhofer Institut : Intelligente Analyse- und Informationssysteme (IAIS): Überblick : Programmiersprachen für Lego Mindstorms NXT. URL: http://www.roberta-home.de/sites/default/files/Vergleich%20Programmiersprachen%20NXT%20v-1.2.pdf. - Stand: 13.01.2012.

[Gam2013] Gambas: Gambas Documentation - What Is Gambas? URL: http://gambasdoc.org/help/doc/whatisgambas?en&view. - Stand: 17.01.2013.

[GH2002] Gediga, Günther ; Hamborg, Kai-Christoph: Evaluation in der Software-Ergonomie. In: Zeitschrift für Psychologie / Journal of Psychology, Volume 210, Number 1/ 2002, 2002, S. 40-57.

[GM2012] Georgi, Wolfgang ; Metin, Ergun: Einführung in LabView : mit 157 Aufgaben ; [Studentenversion inklusive]. München: Fachbuchverlag Leipzig im Carl-Hanser-Verlag, 2012. ISBN 3-446-42386-9, 978-3-446-42386-2.

[GVN1994] Gajski, Daniel D. ; Vahid, Frank ; Narayan, Sanjiv ; Gong, Jie: Specification and design of embedded systems. Englewood Cliffs, NJ: PTR Prentice Hall, 1994. ISBN 0-13-150731-1.

[HB2009] Hutton, Mike ; Betz, Vaughn: FPGA Synthesis and Physical Design. In: Zurawski, Richard (Hrsg.): Embedded systemes handbook : embedded systems design and verification. Boca Raton [u.a.]: CRC Press Taylor & Francis Group, 2009. ISBN 978-1-4398-0755-2. S. (17-01)-(17-34).

[Hea2002] Heath, Steve: Embedded systems design. Oxford: Butterworth-Heinemann, 2002. ISBN 0-7506-5546-1.

[Hei2011] Heinen, Nike: Fokus: Technikspielzeug. In: Technology Review, Volume 12/2011, 2011.

[Hit2011] HiTechnic: NXT SuperPro Prototype Board (SPR2010). URL: http://www.hitechnic.com/cgi-bin/commerce.cgi?preadd=action&key=SPR2010. - Stand: 13.01.2012.

[Ise2008] Isermann, Rolf: Mechatronische Systeme : Grundlagen. 2. Auflage, Berlin [u.a.]: Springer Verlag, 2008. ISBN 978-3-540-32336-2.

[ISO19505-1-2012] Norm, ISO Internationale Organisation für Normung: ISO/IEC 19505-1. Information technology - Object Management Group Unified Modeling Language (OMG UML) - Part 1: Infrastructure, Beuth Verlag GmbH, 2012.

[ISO/IEC19501-2005] Regelwerk, ISO Internationale Organisation für Normung, IEC Internationale Elektrotechnische Kommission: ISO/IEC 19501. Informationstechnik - Offene verteilte Verarbeitung - Vereinheitlichte Modellierungssprache (UML) Version 1.4.2, 2005.

[Jan2010] Janschek, Klaus: Systementwurf mechatronischer Systeme : Methoden - Modelle - Konzepte. Heidelberg [u.a]: Springer Verlag, 2010. ISBN 978-3-540-78876-8.

[Joh1993] Johannsen, Gunnar: Mensch-Maschine-Systeme. Berlin [u.a.]: Springer Verlag, 1993. ISBN 3-540-56152-8.

[Jun2000] Jung, Matthias: Ein Generator zur Entwicklung visueller Sprachen. Universität Paderborn, Fachbereich Mathematik-Informatik, Paderborn, Dissertation, 2000.

[Kie1997] Kieras, David E.: A Guide to GOMS model usability evaluation using NGOMSL. In: Helander, Martin G. ; Landauer, Thomas K. ; Prabhu, Prasad V. (Hrsg.): Handbook of Human-Computer Interaction. Amsterdam: Elsevier, 1997. ISBN 0-444-81862-6. S. 733-766.

[Kol2012] Kolodziej, Daniela: Fachkräftemangel in Deutschland - Statistiken, Studien und Strategien. Deutscher Bundestag, WD 6: Arbeit und Soziales, Infobrief, Akteinzeichen WD 6 – 3010-189/11, 2012.

[Kös2005] Köster, Yvonne: Technologiegestützte Lehr- und Lernmöglichkeiten für die Schule : Ein Beitrag zur Didaktik im gemeinsamen Unterricht der Haupt-, Real- und Gesamtschule. Universität zu Köln, Heilpädagogische Fakultät, Köln, Dissertation, 2005.

[KR1997] Kantner, Laurie ; Rosenbaum, Stephanie: Usability Studies of WWW Sites : Heuristic Evaluation vs. Laboratory Testing. In: Smart, Karl (chairman) (Hrsg): SIGDOC '97 Proceedings of the 15th annual international conference on Computer documentation, Snowbird, UT, USA, 19.-22.10.1997, ACM, New York, NY, USA. ISBN 0-89791-861-4. S. 153-160.

[KRT2010] Kaster, Annette ; Ruckert, C. ; Tappert, E.: Benutzerzentrierte Gestaltung von Benutzungsschnittstellen für komplexe Führungssysteme auf schwimmenden Plattformen. In: VDI Wissensforum, Gesellschaft Mess- und Automatisierungstechnik (Hrsg): 5. VDI Fachtagung - Useware 2010VDI-Berichte Nr. 2099, Baden-Baden, Germany, 13.-.14.10.2010, VDI-Verlag GmbH, Düsseldorf. ISBN 978-3-18-092099-3. S. 237-246.

[Las2012] Laser & Co. Solutions GmbH: Mikrocontroller-Programmierung mit myAVR - Einsteigersets, Entwicklungsboards, Software. URL: http://myavr.de/. - Stand: 19.11.2012.

[Leg2011a] The LEGO Group: 8547 LEGO® MINDSTORMS® NXT 2.0. URL: http://mindstorms.lego.com/en-us/products/default.aspx. - Stand: 16.11.2011.

[Leg2011b] The LEGO Group - LEGO education: 7+ LEGO(R) Education WeDo(TM). URL: http://education.lego.com/en-gb/preschool-and-school/lower-primary-4-7/7plus-lego-education-wedo/. - Stand: 19.12.2011.

[Leg2011c] The LEGO Group: MINDSTORMS : What is in the box? URL: http://mindstorms.lego.com/en-us/history/default.aspx. - Stand: 13.01.2012.

[LP2009] Lavagno, Luciano ; Passerone, Claudio: Design of Embedded Systems. In: Zurawski, Richard (Hrsg.): Embedded systemes handbook : embedded systems design and verification. Boca Raton [u.a.]: CRC Press Taylor & Francis Group, 2009. ISBN 978-1-4398-0755-2. S. (02-01)-(02-26).

[Lpe2006] LPE Technische Medien GmbH: ROBOLAB 2.9 : Lego Mindstorms NXT. URL: http://www.nxt-in-der-schule.de/lego-mindstorms-education-nxt-system/nxt-software/robolab-2.9. - Stand: 13.01.2012.

[LW1997] Lewis, Clayton ; Wharton, Cathleen: Cognitive Walkthroughs. In: Helander, Martin G. ; Landauer, Thomas K. ; Prabhu, Prasad V. (Hrsg.): Handbook of Human-Computer Interaction. Amsterdam: Elsevier, 1997. ISBN 0-444-81862-6. S. 717-732.

[Mar2006] Marwedel, Peter: Embedded system design. 2. Edition, Dordrecht: Springer Verlag, 2006. ISBN 0-387-29237-3.

[Mar2008] Maring, Matthias: Mensch-Maschine-Interaktion : Steuerbarkeit - Verantwortbarkeit. In: Hubig, Chrsistoph ; Koslowski, Peter (Hrsg.): Maschinen, die unsere Brüder werden : Mensch-Maschine-Interaktion in hybriden Systemen. Paderborn [u.a.]: Wilhelm Fink Verlag, 2008. ISBN 3-7705-4593-1. S. 113-129.

[Mat2011a] The MathWorks: Simulink : Simulation und Model-Based Design. URL: http://www.mathworks.com/products/simulink/. - Stand: 10.01.2012.

[Mat2011b] Matrix Multimedia: Flowcode graphical programming languages for microcontrollers : rapid electronic development kits. URL: http://www.matrixmultimedia.com/flowcode.php. - Stand: 10.01.2012.

[Mat2012a] The Mathworks, Inc.: Simulink Support for Microchip dsPIC Microcontrollers. URL: http://www.mathworks.de/academia/microchip/. - Stand: 09.01.2012.

[Mat2012b] Matrix Multimedia: FlowCode Data Sheet. URL: http://www.matrixmultimedia.com/resources/files/datasheets/Flowcode5 Booklet-2.pdf. - Stand: 07.11.2012.

[Mat2012c] The Mathworks, Inc.: MathWorks Deutschland - MATLAB Coder mit Simulink Coder und Embedded Coder - MATLAB Coder. URL: http://www.mathworks.de/products/matlab-coder/description5.html. - Stand: 15.11.2011.

[May2008] Mayhew, Deborah J.: The usability engineering lifecycle : a practitioner's handbook for user interface design. San Francisco, Calif. [u.a.]: Morgan Kaufmann, 2008. ISBN 1-558-60561-4.

[Mel1999] Melezinek, Adolf: Ingenieurpädagogik : Praxis der Vermittlung technischen Wissens. 4. neubearbeitete Auflage, Wien [u.a.]: Springer Verlag, 1999. ISBN 3-211-83305-6.

[Mic2011a] Microchip: MPLAB Integrated Development Environment. URL: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002. - Stand: 10.01.2012.

[Mic2011b] Microchip: MPLAB ICD 3 In-Circuit Debugger. URL: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en537580. - Stand: 10.01.2012.

[Mic2013a] Microchip: PIC18F2520. URL: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010277. - Stand: 02.01.2013.

[Mic2013b] Microsoft Corporation: Visual Studio 2012 Startseite. URL: http://www.microsoft.com/germany/visualstudio/. - Stand: 02.01.2013.

[Mor2001] Morton, Todd D.: Embedded microcontrollers. Upper Saddle River, NJ [u.a.]: Prentice Hall, 2001. ISBN 0-13-907577-1.

[Müt2009] Mütterlein, Bernward: Handbuch für die Programmierung mit LabVIEW : [mit Studentenversion LabVIEW 8]. Heidelberg: Spektrum Akademischer Verlag, 2009. ISBN 978-3-8274-2337-5.

[Nat2011] National Instruments Corporation: LEGO MINDSTORMS, Powered by NI LabVIEW. URL: http://www.ni.com/company/education/mindstorms.htm. - Stand: 13.01.2012.

[Nat2012] National Instruments Corporation: NI LabVIEW: Höhere Produktivität für Ingenieure und Wissenschaftler. URL: http://www.ni.com/labview/d/. - Stand: 14.11.2012.

[NB1999] Noyes, Janet M. ; Baber, Christopher: User-centered design of systems. In: Paul, Ray J. ; Thomas, Peter J. ; Kuljis, Jasna (Hrsg.): Applied Computing. London [u.a.]: Springer-Verlag, 1999. ISBN 3-540-76007-5.

[NGL2000] Neumann, Peter ; Grötsch, Eberhard E. ; Lubkoll, Christoph ; Simon, René: SPS-Standard: IEC 61131 : Programmierung in verteilten Automatisierungssystemen. München [u.a.]: Oldenbourg Verlag, 2000. ISBN 3-486-27005-2.

[Nie1994] Nielsen, Jakob: Estimating the number of subjects needed for a thinking aloud test. In: International Journal of Human-Computer Studies, Volume 41, Issue 3, 1994, S. 385-397.

[Nie2005] Nielsen, Jakob: Usability for the masses. In: Journal of Usability Studies, Volume 1, Issue 1, 2005, S. 2-3.

[Oes2012] Oestereich, Bernd: Analyse und Design mit der UML 2.5 : objektorientierte Softwareentwicklung ; [inkl. Poster mit UML-Notationsübersicht]. 10., aktualisierte und erweiterte Auflage, München: Oldenbourg Wissenschaftsverlag GmbH, 2012. ISBN 978-3-486-71667-2.

[Ora2012] Oracle Corporation: Was ist Java? URL: http://www.java.com/de/download/whatis_java.jsp. - Stand: 23.11.2011.

[Pat2000] Paternò, Fabio: Model-Based Design and Evaluation of Interactive Applications. In: Paul, Ray J. ; Thomas, Peter J. ; Kuljis, Jasna (Hrsg.): Applied Computing. 1. Auflage, London [u.a.]: Springer Verlag, 2000. ISBN 1-85233-155-0.

[PDB2009] Passerone, R. ; Damm, W. ; Ben Hafaiedh, I. ; Graf, S. ; Ferrari, A. ; Mangeruca, L. ; Benveniste, A. ; Josko, B. ; Peikenkamp, T. ; Cancila, D. ; Cuccuru, A. ; Gerard, S. ; Terrier, F. ; Sangiovanni-Vincentelli, A.: Metamodels in Europe: Languages, Tools, and Applications. In: Design & Test of Computers, IEEE, Volume 26, Issue 3, 2009, S. 38-53.

[Pec2008] Peckol, James K.: Embedded systems : a contemporary design tool. Hoboken, NJ.: Wiley, 2008. ISBN 0-471-72180-8, 978-0-471-72180-2 (cloth).

[PSS2008] Parab, Jivan S. ; Shinde, Santosh A. ; Shelake, Vinod G. ; Kamat, Rajanish K. ; Naik, Gourmish M.: Practical Aspects of Embedded System Design using Microcontrollers. Dordrecht: Springer, 2008. ISBN 1-402-08392-0, 978-1-402-08392-1*Gb.

[Ren1996] Renkl, Alexander: Vorwissen und Schulleistung. In: Möller, Jens ; Köller, Olaf (Hrsg.): Emotionen, Kognitionen und Schulleistung. Weinheim: Beltz, Psychologie-Verl.-Union. ISBN 3-621-27339-5. S. 175-190.

[Rod2012] Roddeck, Werner: Einführung in die Mechatronik - Mit 494 Abbildungen - STUDIUM. 4., überarbeitete Auflage, Wiesbaden: Vieweg+Teubner Verlag | Springer Fachmedien Wiesbaden GmbH, 2012. ISBN 978-3-8348-1622-1.

[RQS2012] Rupp, Chris ; Queins, Stefan ; SOPHISTen, die: UML 2 glasklar : Praxiswissen für die UML-Modellierung. 4., aktualisierte und erweiterte Auflage, München: Carl Hanser Verlag, 2012. ISBN 978-3-446-43057-0.

[RSS1994] Rauterberg, Matthias ; Spinas, Philipp ; Strohm, Oliver ; Wich, Eberhard ; Waeber, Daniel: Benutzerorientierte Software-Entwicklung : Konzepte, Methoden und Vorgehen zur Benutzerbeteiligung. In: Ulich, Eberhard (Hrsg.): Mensch, Technik, Organisation ; Band 3. Zürich: vdf, Hochschulverlag an der ETH Zürich [u.a.], 1994. ISBN 3-519-02159-5.

[SA2011] Schmid, Marco ; Ahrends, Stephan: Grafisches System Design im Embedded-Bereich : Plattformübergreifend entwickeln - vom PXI über Singleboard Computer und FPGA zum Microcontroller. In: Jamal, Rahman ; Heinze, Ronald (Hrsg.): Virtuelle Instrumente in der Praxis 2011 : Begleitband zum 16. VIP-Kongress. Berlin, Offenbach: VDE Verlag GmbH. ISBN 978-3-8007-3329-3. S. 208-216.

[SB2011] Sarodnick, Florian ; Brau, Henning: Methoden der Usability Evaluation : Wissenschaftliche Grundlagen und praktische Anwendung. In: Bamberg, Eva ; Mohr, Gisela ; Rummel, Martina (Hrsg.): Wirtschaftspsychologie in Anwendung. 2., überarb. und aktualisierte Aufl., Bern: Verlag Hans Huber, Hogrefe AG, 2011. ISBN 978-3-456-84883-9.

[Sch1998] Schiffer, Stefan: Visuelle Programmierung : Grundlagen und Einsatzmöglichkeiten. Bonn [u.a.]: Addison-Wesley, 1998. ISBN 978-3-8273-1271-6.

[Sen2013] Sensirion AG: Humidity Sensor SHT11: Sensirion AG. URL: http://www.sensirion.com/en/products/humidity-temperature/humidity-sensor-sht11/. - Stand: 02.01.2013.

[She1992] Sheridan, Thomas B.: Telerobotics, Automation, and Human Supervisory Control. Cambridge, Massachusetts: The MIT Press, 1992. ISBN 0-262-19316-7.

[Sie2012] Siemens AG: SIMATIC STEP 7 Software - Software für SIMATIC Controller - Siemens. URL: http://www.automation.siemens.com/mcms/simatic-controller-software/de/step7/Seiten/Default.aspx. - Stand: 20.11.2012.

[SS2007] Sumathi, Sai ; Surekha, Paneerselvam: LabVIEW based Advanced Instrumentation Systems : with 34 tables. Berlin [u.a.]: Springer-Verlag, 2007. ISBN 3-540-48500-7, 978-3-540-48500-1.

[SSS2012a] 3S-Smart Software Solutions GmbH: Das CoDeSys Geräteverzeichnis: alles was sie brauchen. URL: http://www.3s-software.com/index.shtml?codesys_dev_dir. - Stand: 20.11.2012.

[SSS2012b] 3S-Smart Software Solutions GmbH: CoDeSys - das IEC 61131-3 SPS-Programmiersystem der Automatisierungstechnik. URL: www.3s-software.com. - Stand: 20.11.2012.

[ST2003] Schweibenz, Werner ; Thissen, Frank: Qualität im Web : benutzerfreundliche Webseiten durch Usability Evaluation. Berlin [u.a.]: Springer, 2003. ISBN 3-540-41371-5.

[Ste2005] Stetter, Rainer: Der Weg zum mechatronischen Engineering : Koordiniertes Engineering auf der Basis mechatronischer Baukasten. In: Technica - Die Fachzeitschrift für die Industrie, 54. Jahrgang, Volume 19/2005, 2005, S. 10-12.

[Tex2012] TEXAS INSTRUMENTS: Code Composer Studio (CCStudio) Integrated Development Environment (IDE) v5 - CCSTUDIO. URL: http://www.ti.com/tool/ccstudio. - Stand: 14.11.2012.

[Val2010] Valvano, Jonathan W.: Introduction to Embedded Systems : Interfacing to the Freescale 9S12. 1st Edition (International Student Edition), Stamford, CT: Cengage Learning, 2010. ISBN 0-495-41138-8, 978-0-495-41138-3.

[VDI2206-2004] Regelwerk, VDI-Gesellschaft Produkt- und Prozessgestaltung: VDI 2206. Entwicklungsmethodik für mechatronische Systeme, Beuth Verlag GmbH, 2004.

[VDI3699-2005] Regelwerk, VDI - Verein Deutscher Ingenieure: VDI/VDE 3699 Blatt 1. Prozessführung mit Bildschirmen : Begriffe, Beuth Verlag GmbH, 2005.

[VDMA66305-2003] Regelwerk, VDMA - Verband Deutscher Maschinen- und Anlagenbau e. V.: VDMA 66305. Bausteine und Schnittstellen der Mikroctechnik, Beuth Verlag GmbH, 2003.

[VIR2002] Vredenburg, Karel ; Isensee, Scott ; Righi, Carol: User-centered design : An Integrated Approach. In: Belady, Les ; Clements, Paul ; Dale, Al ; Freeman, Peter ; Krasner, Herb ; Musa, John ; Otter-Nickerson, Betty ; Pfleeger, Shari L. ; Wasserman, Tony (Hrsg.): Software Quality Institute Series. Upper Saddle River, NJ [u.a.]: Prentice Hall PTR, 2002. ISBN 0-13-091295-6.

[Win2000] Winter, Andreas: Referenz-Metaschema für visuelle Modellierungssprachen. 1. Aufl., Wiesbaden: Dt. Univ.-Verlag, 2000. ISBN 3-8244-2140-2.

[WRL1994] Wharton, Cathleen ; Rieman, John ; Lewis, Clayton ; Polson, Peter: The Cognitive Walkthrough Method : A Practitioner's Guide. In: Nielsen, Jakob ; Mack, Robert L. (Hrsg.): Usability Inspection Methods. New York: John Wiley & Sons, Inc., 1994. ISBN 0-471-01877-5. S. 105-140.

[Wüs2011] Wüst, Klaus: Mikroprozessortechnik : Grundlagen, Architekturen, Schaltungstechnik und Betrieb von Mikroprozessoren und Mikrocontrollern ; mit 195 Abbildungen und 44 Tabellen. 4., aktualisierte und erw. Aufl., Wiesbaden: Vieweg+Teubner Verlag | Springer Fachmedien, 2011. ISBN 978-3-8348-0906-3.

[Züh2004] Zühlke, Detlef: Useware-Engineering für technische Systeme. 1. Auflage, Berlin [u.a.]: Springer Verlag, 2004. ISBN 3-540-20647-7.